TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TEEMU JÄMSÄ

DNS-BASED AUTHENTICATION OF NAMED ENTITIES

Master of Science Thesis

Examiner: Professor Jarmo Harju
Supervisors: M.Sc. Joona Kannisto and M.Sc. Aleksi Suhonen
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 4 November 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
JÄMSÄ, TEEMU: DNS-based Authentication of Named Entities
Master of Science Thesis, 53 pages
June 2014
Major: Communication Networks and Protocols
Examiner: Professor Jarmo Harju
Keywords: DANE, TLSA, DNSSEC, Encrypted Email

Public Key Infrastructure (PKI) has turned out to be useful when two parties negotiate about a shared secret in order to establish an encrypted connection between them. To verify the public key, a certificate is used. The certificate is issued by a public, generally trusted third party Certificate Authority (CA). Usually, the web browsers have a list of trusted CAs. It is a well-known problem that the number of security risks increases when the number of CAs grows. A compromised CA can, by an attacker's malicious action or by a human error, issue a trusted certificate to a party who does not own the domain.

The purpose of this Master of Science Thesis is to research the applications of the DANE protocol, which is standardized by the IETF. The research question is, how to validate a target receiver while negotiating the encrypted connection. Special focus is on the secure email system. The DANE protocol makes use of the existing Domain Name System (DNS) and its Security Extensions (DNSSEC).

This Master of Science Thesis begins with a theoretical part, where the technical background and current techniques are introduced. The DANE protocol and its features are also considered in this chapter. The latter part considers the method in practice, and describes how DANE can be used for the certificate verification instead of CA.

The testing phase proves that the deployment of DANE is not complex and the increase of delay and traffic are not significant. DANE provides the needed association between the DNSSEC's chain of trust and the received certificate.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
JÄMSÄ, TEEMU: DNS-based Authentication of Named Entities
Diplomityö, 53 sivua
Kesäkuu 2014
Pääaine: Tietoliikenneverkot ja protokollat
Tarkastaja: professori Jarmo Harju
Avainsanat: DANE, TLSA, DNSSEC, Salattu sähköposti

Salattuja tietoliikenneyhteyksiä luotaessa ja yhteistä salausavainta neuvoteltaessa voidaan käyttää hyväksi julkisen avaimen järjestelmää. Jotta toisen osapuolen oikeellisuus voidaan todentaa, sen julkinen avain verifioidaan sertifikaatilla, jonka myöntää julkinen, yleisesti luotettu varmentaja. Tietokoneiden selaimet yleensä automaattisesti luottavat yleisesti tunnettuihin varmentajiin. Varmentajien lukumäärän kasvaessa myös tietoturvariski kasvaa. Tietomurron kohteeksi joutunut varmentaja voi hyökkääjien toimesta, tai vaikka vain epähuomiossa, julkaista luotetun sertifikaatin mille tahansa taholle, jolle se ei kuulu.

Tässä diplomityössä tutkitaan IETF:n spesifioiman DANE-protokollan soveltuvuutta salatun tietoliikenneyhteyden kohteen todentamiseen. Erityisenä tarkastelukohteena ovat salatun sähköpostiyhteyden osapuolet. DANE-protokolla hyödyntää jo valmiiksi rakennettua, hierarkkista nimipalvelujärjestelmää (DNS) ja erityisesti sen turvallisuuslaajennuksen (DNSSEC) avulla digitaalisesti allekirjoitettuja tietueita.

Diplomityön alkuosassa käsitellään tutkimuksen teoreettinen osuus, jossa käydään läpi teknistä taustaa ja menetelmiä, jotka ovat nykypäivänä käytössä. Osiossa esitellään myös DANE-protokolla ja siihen liittyvät ominaisuudet. Diplomityön jälkimmäisessä osassa suoritetaan käytännönläheinen testaus, jossa DANE-protokollaa hyödynnetään sertifikaattien todentamisessa julkisten varmentajien sijaan.

Testaus osoittaa, että DANE-protokollan käyttöönotto on helppoa olemassa olevaan DNSSECiin tukeutuen, eikä viive tai pakettiliikenteen määrä kasva kohtuuttomasti. DANE-protokolla luo tarvittavan siteen DNSSECin luottamusketjun ja kohteen sertifikaatin välille ja parantaa näin järjestelmän tietoturvaa.

PREFACE

This Master of Science Thesis was written for the Department of Pervasive Computing at Tampere University of Technology during the academic year 2013 - 2014. This thesis was examined by Professor Jarmo Harju and supervised by M.Sc. Joona Kannisto and M.Sc. Aleksi Suhonen.

Tampere
April 17, 2014

Teemu Jämsä

CONTENTS

ABBREVIATIONS

AD	Authenticated Data. A DNSSEC message header bit.
CA	Certificate Authority. An organization who issues digital certificates.
ccTLD	Country Code Top Level Domains. A domain name space for a country.
CD	Checking Disabled. A DNSSEC message header bit.
DANE	Domain Name System based Authentication of Named Entities. A protocol, which provides an association between a domain name and a certificate.
DER	Distinguished Encoding Rules. An encoding syntax.
DNS	Domain Name System. A naming system for Internet.
DNSKEY	Domain Name System Public Key. A DNSSEC resource record type.
DNSSEC	Domain Name System Security Extensions. A security extension for DNS.
DO	Domain Name System Security Extensions OK.
DS	Delegation Signer. A DNSSEC resource record type.
EDNS	Extension Mechanisms for Domain Name System.
gTLD	Generic Top Level Domain. A general term for .com-, .net-, .org- and .gov-domains.
ICANN	Internet Corporation for Assigned Names and Numbers.
IETF	Internet Engineering Task Force. The standardization organization.
IoT	Internet of Things.
IP	Internet Protocol. A protocol on the network layer.
KSK	Key Signing Key. One of the public keys for DNSSEC.
M2M	Machine to Machine. A connection between similar devices.
MIME	Multipurpose Internet Mail Extension. A standard for email extensions.
MTA	Message Transfer Agent. A software for email transmission.
MX	Mail Exchange. A DNS record for email servers.
NS	Name Server. A DNS name server.
NSD	Name Server Daemon. A DNS name server software.
NSEC	Next Secure. A DNSSEC resource record type.
PKI	Public Key Infrastructure. An arrangement for public keys.
PKIX	Public Key Infrastructure (X.509). PKI, which uses X.509.
PMI	Privilege Management Infrastructure.
PSK	Pre-Shared Key. A symmetric secret key.
RDATA	Resource record Data.

RFC	Request for Comments. Internet protocol standards published by IETF.
RR	Resource Record. Contains specific information about an object.
RRSIG	Resource Record Digital Signature. A DNSSEC resource record type.
S/MIME	Secure/Multipurpose Internet Mail Extensions. A standard for encryption and signing of MIME.
SMIMEA	Not an acronym. A resource record of DANE.
SMTP	Simple Mail Transfer Protocol. A protocol for email transmission.
TCP	Transmission Control Protocol. A transport layer protocol.
TLD	Top Level Domain. A domain name space level below the root zone.
TLS	Transport Layer Security. A cryptographic protocol.
TLSA	Not an acronym. A resource record of DANE.
TTL	Time-To-Live. A pre-set lifetime.
TXID	Transaction Identifier. A 16 bit identification number for DNS.
UDP	User Datagram Protocol. A transport layer protocol.
X.509	Not an acronym. A standard for a PKI and PMI.
ZSK	Zone Signing Key. One of the public keys for DNSSEC.

1 INTRODUCTION

The communication between the various types of applications over the insecure Internet has led to a regrettable situation where the traffic is monitored, spoofed and eavesdropped and data packets are being tampered or forged. The security between the parties can be improved by using the Transport Layer Security (TLS), which provides a secure connection by encrypting the channel.

The provision of the security made by TLS is based on the mutual encryption, where the secrecy of the channel is provided, for example, by using secret, private keys of the parties. The provided channel encryption is as strong as the used cipher suite, which includes the authentication, encryption and algorithms. A weak secret can be revealed, which leads to the case that the encryption of the channel could be compromised and there will be no security anymore. The worst case scenario is that this malicious action can be completely invisible to the original connection parties.

The authentication of TLS is based on the Public Key Infrastructure (PKI) and one problem is, how to trust a key issued by another party. TLS has solved this problem by using certificates and Certificate Authorities (CA). The CAs are the trusted third party organizations, who sign the certificates to the TLS servers and services, and the TLS clients can verify those certificates by using the published public key of the CA.

But unfortunately, a trusted PKI system can only be as strong as the weakest CA. This enables one of the biggest security problems. When one CA is compromised, it can issue a replacement certificate to any other domain name. This malicious action can cause serious damage to millions of users.

The Domain Name System Security Extensions (DNSSEC) improves the reliability and the data integrity of the Domain Name System (DNS) by building a chain of trust from the root by signing the untrusted keys associated to the domain name zones below it, so called child zones. DNSSEC is also based on public keys like the TLS, but instead of hundreds of different CAs, hierarchical DNSSEC has to trust in essence only the root.

The DNS-based Authentication of Named Entities (DANE) provides an association between the reliable DNSSEC infrastructure and the TLS certificate verified by the CA. The DANE protocol allows the domain name administrator to store the public key of the TLS server to the DNS data. This associated public key must match the public key in the TLS certificate. This prevents fake certificates issued by compromised CAs.

This Master of Science Thesis introduces DANE and the reasons, which have led to its invention. The DANE-validated TLS encryption between email servers was

selected as a main study. Chapter 2 will go through the theoretical and technical background of the topic. At first, TLS and its functionality are presented. Then DNS, its functionality and vulnerabilities are very briefly presented. Thirdly, DNSSEC, its functionality, building of the chain of trust and the DNSSEC's challenges are also briefly presented. At last, the problems and the needs are considered before entering the next chapter. In Chapter 3, DANE is fully introduced. The chapter goes through the details of DANE, the TLSA record format and the TLSA record rollover. Some examples are also presented in Chapter 3. DANE and TLSA records with encrypted SMTP are described in more specifically. Chapter 4 introduces the test environment, which was built for testing the encrypted email transmissions with DANE TLSA records. These records were received via DNSSEC validated DNS responses. The chapter will go through the basic configuration and the testing phase. The test results found during the testing phase are considered and analyzed in Chapter 5. The conclusions of this thesis are presented in Chapter 6.

2 BACKGROUND

This chapter presents the theoretical and technical background of the subject. It goes through the techniques and methods which are nowadays widely used to provide a secure connection between two parties, for example, a bank's web server and a client's home computer.

2.1 Transport Layer Security

The Transport Layer Security (TLS) is a cryptographic protocol which uses symmetric secrecy to provide communication security over the Internet. In this chapter, the functionality of TLS is briefly presented. The full functionality of the TLSv1 is explained in the Internet Engineering Task Force (IETF) RFC 2246 standard in January 1999, the TLS version 1.1 in the IETF RFC 4346 standard in April 2006 and the current, upgraded TLS version 1.2 in the IETF RFC 5246 standard in August 2008.

2.1.1 Functionality

The goal of the TLS protocol is to build a secure connection between a client and a server over the insecure Internet. TLS is a key exchange protocol, which is divided into the two sub protocols: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol lies on top of the transport layer and it is used to encapsulate higher-level protocols. TLS uses the TLS Handshake Protocol to authenticate the session parties and to negotiate the session cipher suites. The authentication is usually made by using the asymmetric Public Key Infrastructure (PKI). The connection is private, and the symmetric secret key, which is then used to encrypt the application data, is privately negotiated for each connection. The generation of the symmetric secret key is made by using the public key of the session party, using the Pre-Shared Key (PSK) or using the both parties' public keys. The handshake overview is shown in Figure 1, where the asterisk indicates optional messages whereas *ChangeCipherSpec* messages are in brackets because they are not actually TLS handshake messages but independent TLS protocol content types. [1]

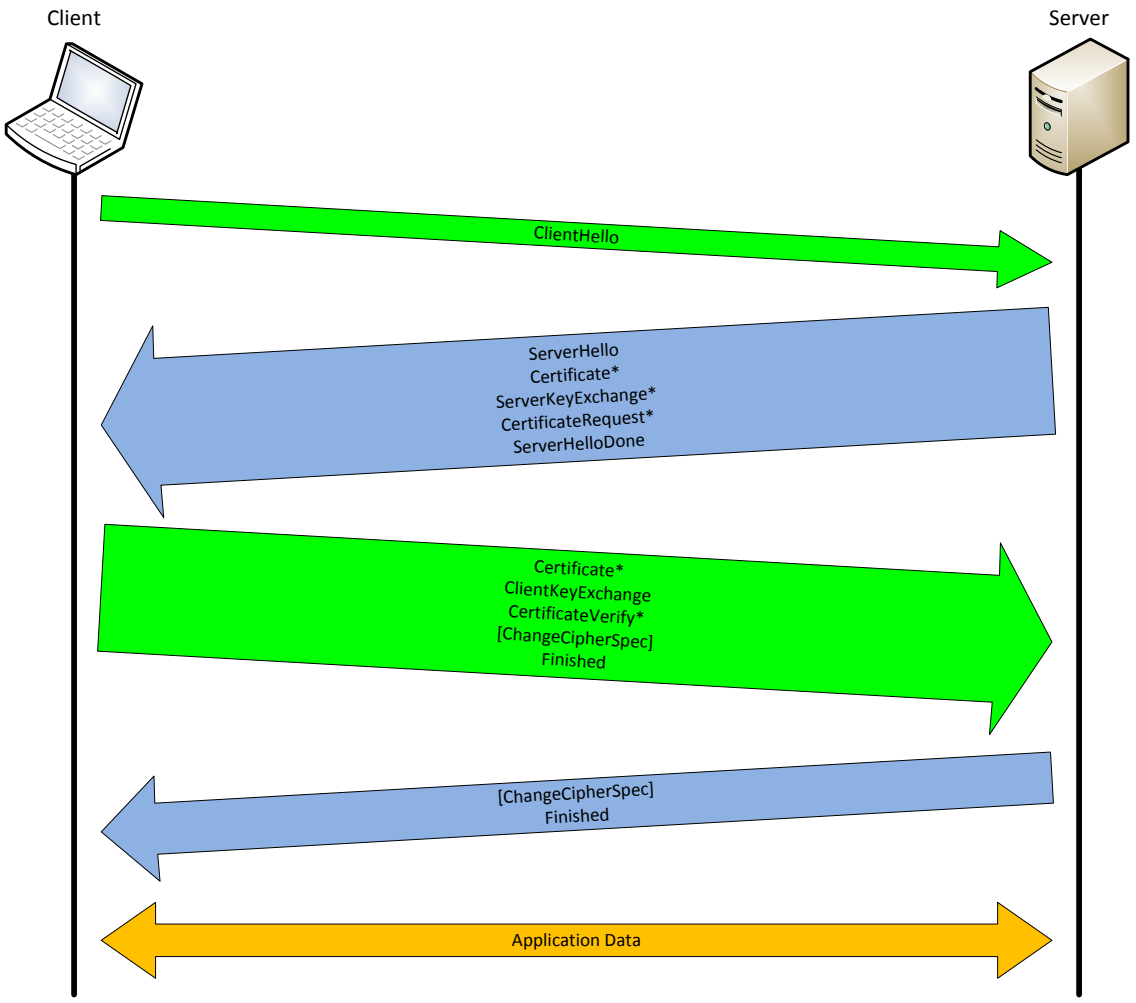

Figure 1. The TLS handshake overview.

The handshake process begins when the client sends its ClientHello message to the server. The *ClientHello* message is used to identify the session and to inform the server about the security enhancement capabilities of the client. [1]

The server will respond to a *ClientHello* message by sending its own *ServerHello* message if it is able to support client's cipher suite, which means a combination of security algorithms. Otherwise it will send a handshake failure alert. Then the server chooses the security policies, such as the cipher suite and the compression method from the list in *ClientHello* message. Immediately after the *ServerHello* message, the server will send a *Certificate* message, which conveys server's certificate information to the client. Only an anonymous negotiation, where the key is pre-shared, will prevent the server from sending this message. Then, in some cases, the server will send the *ServerKeyExchange* message. This message contains extra information, which allows the client to exchange the premaster secret. The server, which is a non-anonymous server, so it had to send the *Certificate* message, could also request a certificate from the client by sending a *CertificateRequest* message. And finally, the server will send a *ServerHelloDone* message, which indicates that this is the end of *ServerHello* and associated

messages. The server has done its part of the key exchange and is now waiting for a client's response. [1]

If the server requested a certificate, the client will first send its own certificate by sending a *Certificate* message. If the client does not have an appropriate certificate, it has to send a *Certificate* message without any actual certificates within. Then the client will send the mandatory *ClientKeyExchange* message. The premaster key is set with this message. If the Certificate message is sent, the client will next send a *CertificateVerify* message to verify its own certificate. At this point, the client will send the *ChangeCipherSpec* message where it informs the server that the following records will be protected. The client ends its part of the handshake by sending a *Finished* message where it verifies that the key exchange was successful. The *Finished* message is the first message, which is protected with the secret just negotiated. Now the client is just waiting for the server's *Finished* message. [1]

The server will also send its *ChangeCipherSpec* and *Finished* messages in that particular order. When the client has received the server's *Finished* message it is ready to send and receive the protected application data over the negotiated connection. [1]

2.1.2 Considerations

The goal of the TLS protocol is to provide a secure and reliable connection between two parties. The negotiation of a shared secret is unavailable to eavesdroppers and attackers in the middle of the connection. The modifications made by attackers can always be detected by the communication parties. The negotiated secret keys are for symmetric encryption and only for that particular connection. [1]

Before the private and reliable connection over the TLS is in use, the client has to solve the location of the target server before it can begin to negotiate the shared secret.

2.2 Domain Name System

The Domain Name System (DNS) is a global, hierarchical naming system for the Internet. It contains mappings between, for example, domain names, Internet Protocol (IP) addresses, text records, mail exchange records and name server information records. Thus, it allows clients to use memorable names, such as www.example.com, rather than IP addresses to find resources on the Internet.

DNS is an IETF standard and its original specifications were presented in the IETF RFC 882 and RFC 883 standards in November 1983. Four years later, in November 1987, they were displaced by the IETF RFC 1034 and RFC 1035 standards. Later on, the IETF has published several additional extension RFCs for DNS. At first in this chapter, the functionality of the DNS is briefly presented. Then, the framework and vul-

nerability are presented before the last part of the chapter, where the results are revised. [2]

2.2.1 Functionality

The structure of DNS is hierarchical and tree-like and its domain name space starts at the root zone which is marked by using a dot symbol. From the top level root, the DNS name space hierarchy is divided into several sub domains as it is shown in Figure 2.

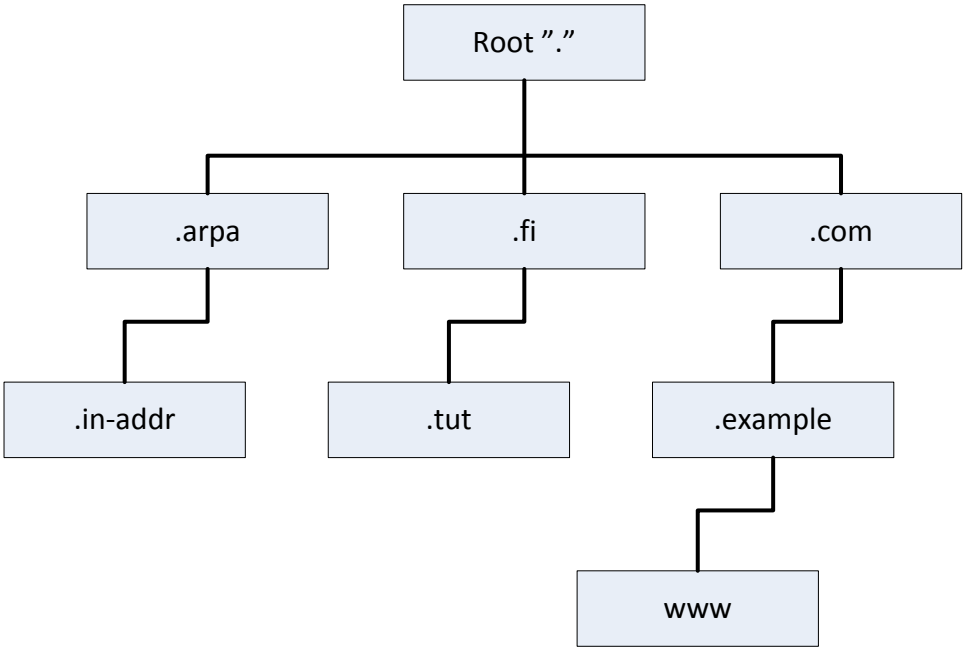

Figure 2. The part of the domain name space tree-like hierarchy.

The .arpa-domain is usually used for reverse mappings that are queries from addresses to hostnames. The .fi-domain is a country code top level domain (ccTLD) for Finland. The .com-domain is a generic TLD (gTLD). The root name server is aware of the domain name spaces below it. Like it is shown in Figure 2, the root knows, for example, the .com-domain which then knows the .example.com-domain which knows the address of the www-server.

If a client wants to connect to the www.example.com, the client's DNS resolver has to resolve the IP address of the www.example.com. The procedure begins with a DNS query where the client's DNS resolver asks about the IP address. The query is sent to the client's domain name server. This is shown in Figure 3.

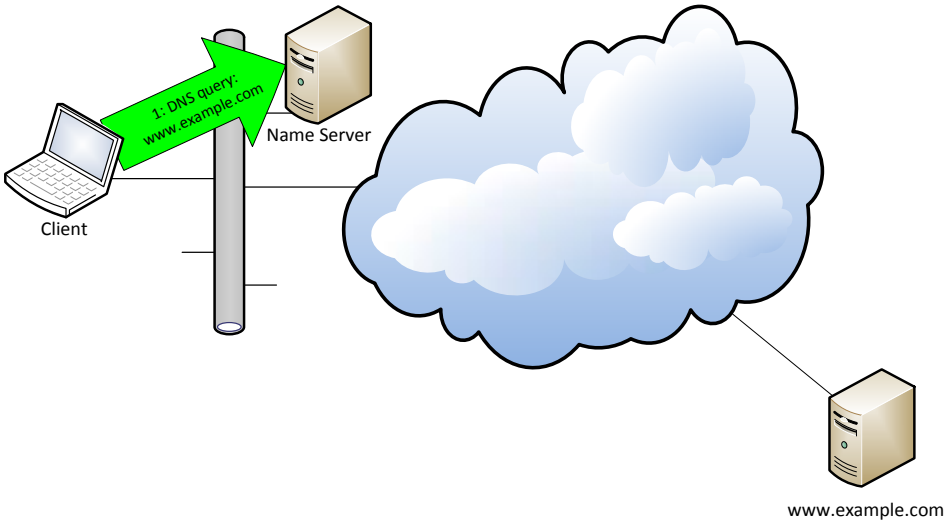

Figure 3. The DNS query from the client's DNS resolver.

If the name server's cache is empty or it does not have any information about www.example.com, it has to ask it from the root server. The IP address of the root server is configured to the name server, so it can always ask at least from it. The root server query is shown in Figure 4.

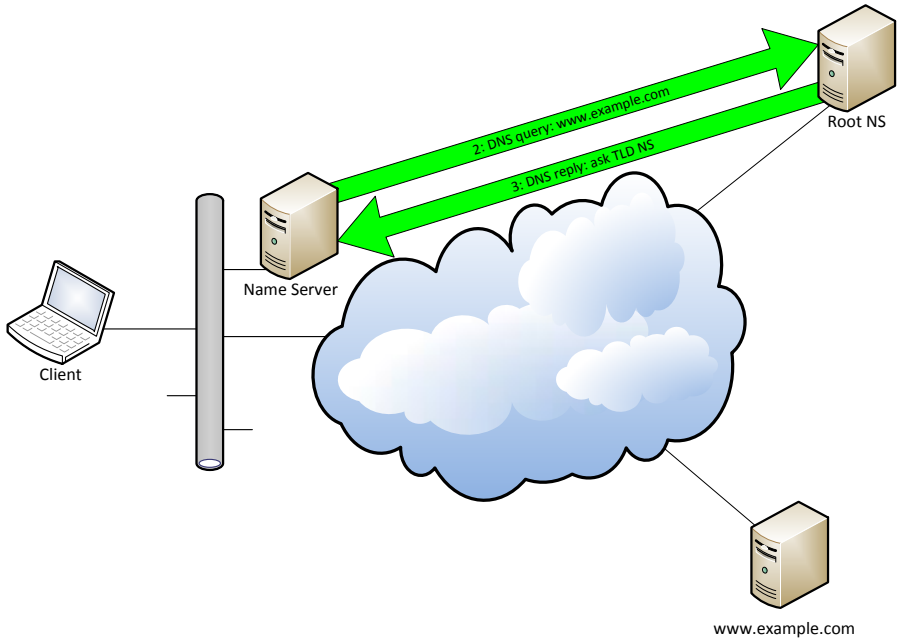

Figure 4. Querying the root name server.

If the queried hostname is valid, the root server knows where the response is found and replies that information to the resolving name server. In Figure 4, the root name server (NS) replies that it does not know the exact IP address of the queried www.example.com, but it knows the IP address of the TLD (also gTLD) NS for .com-domain name space which should know the target IP address. Next, the resolving name server sends its query to the TLD NS. This is shown in Figure 5.

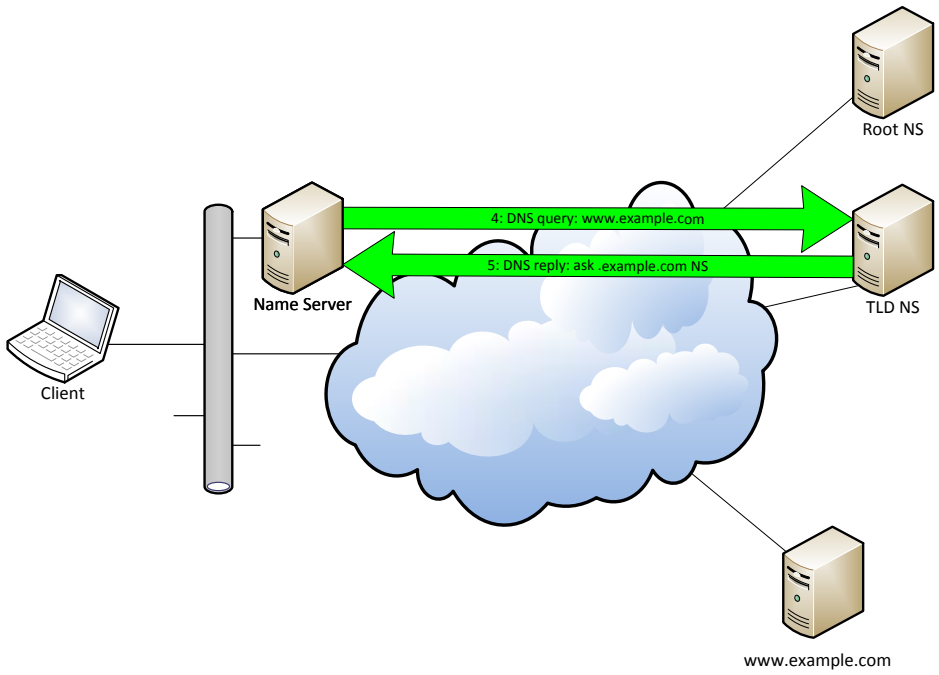

Figure 5. Querying the TLD name server.

In Figure 5, the resolving name server is now querying the .com-domain's name server with the same query. Even if the query is still valid, the TLD NS does not know the exact IP address for the queried www.example.com, but it knows the IP address of the .example.com-domain's name server. TLD NS informs the resolving name server with that information. Then the resolving name server sends its query to the name server of the .example.com-domain. This is shown in Figure 6.

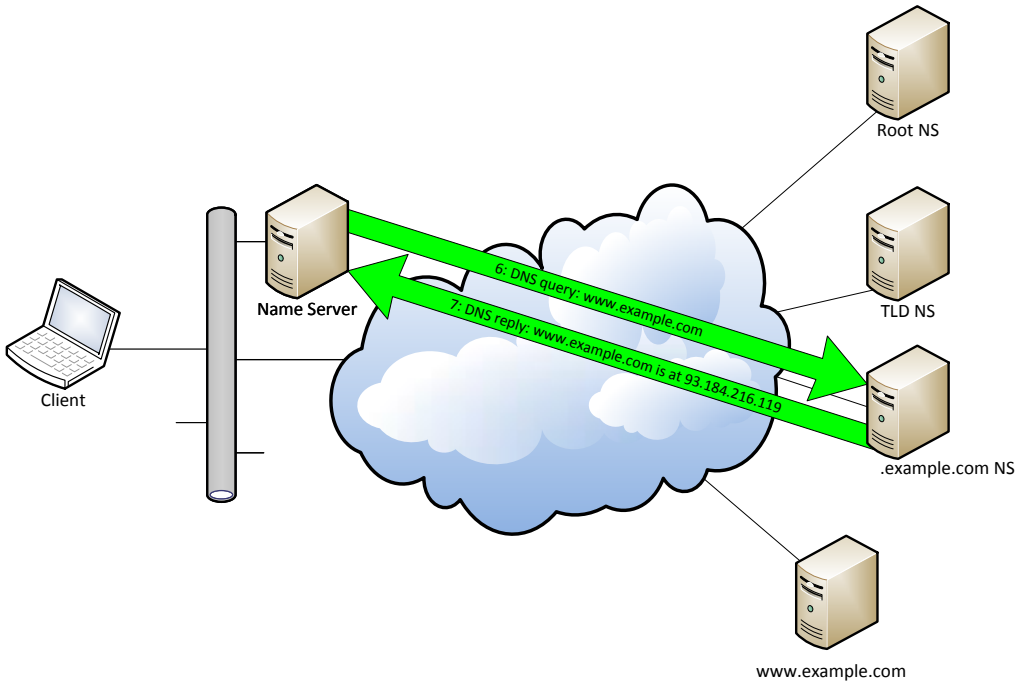

Figure 6. Querying the target domain's name server.

In Figure 6, the resolving name server sends the same DNS query to the target domain's name server which knows the exact IP address of the queried www server. It sends its response which informs the resolving name server that the IP address of the queried www.example.com is 93.184.216.119. Next, the resolving name server will end this query by sending the reply to the client's resolver. This is shown in Figure 7.

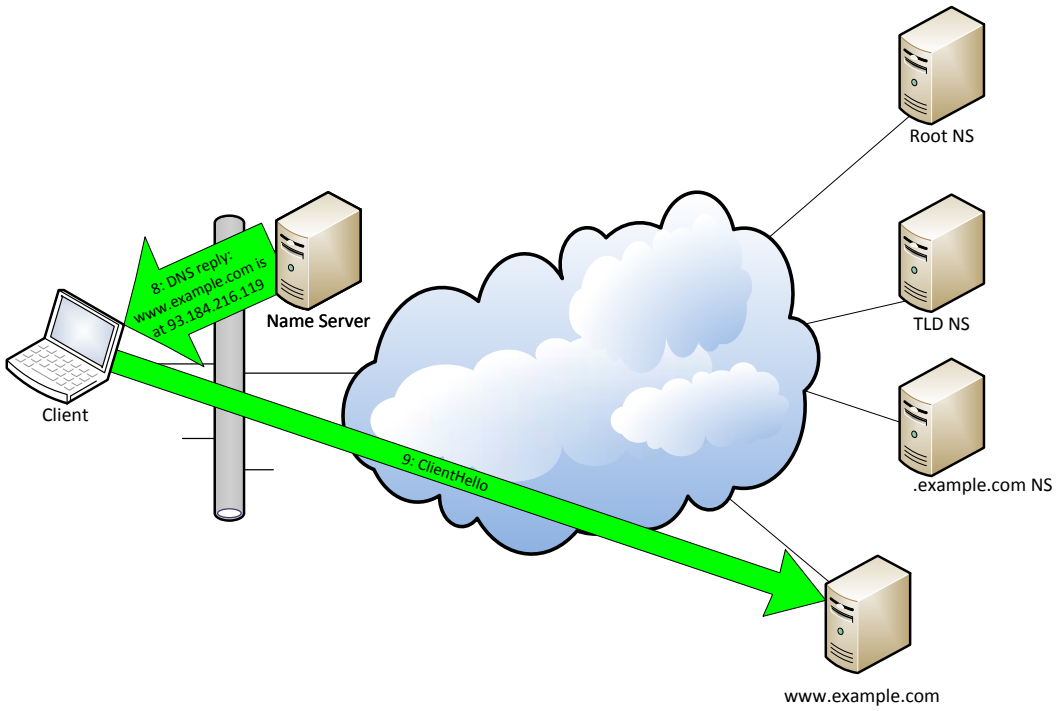

Figure 7. Finishing the query.

In Figure 7, the resolving name server sends its reply to the client's resolver and in that reply it informs that the IP address of the www.example.com is 93.184.216.119 and finally stores this mapping to its cache for a pre-set lifetime, Time To Live (TTL) value. Now the resolving name server has resolved the query by using a non-recursive (or an iterative) resolving method. The original query given by the client's resolver was recursive.

The client's resolver has now received the response for the original query and has the IP address of the web server of the example.com-domain. The client's application, such as web browser begins to negotiate the connection with the web server. [2]

2.2.2 Weaknesses

In a normal case, the client is not suspicious about the reply received from the resolving name server. The client assumes that the response can be relied on. But DNS does not provide strong authentication for the responses. There are several malicious attacks which can spoil or modify the correct DNS resource records and cause that the incorrect information is transmitted to the clients.

In the DNS Cache Poisoning attack, an attacker does an active attack, which causes the resolving name server to approve an incorrect DNS data and to store it to its cache. This malicious method is presented in Figure 8.

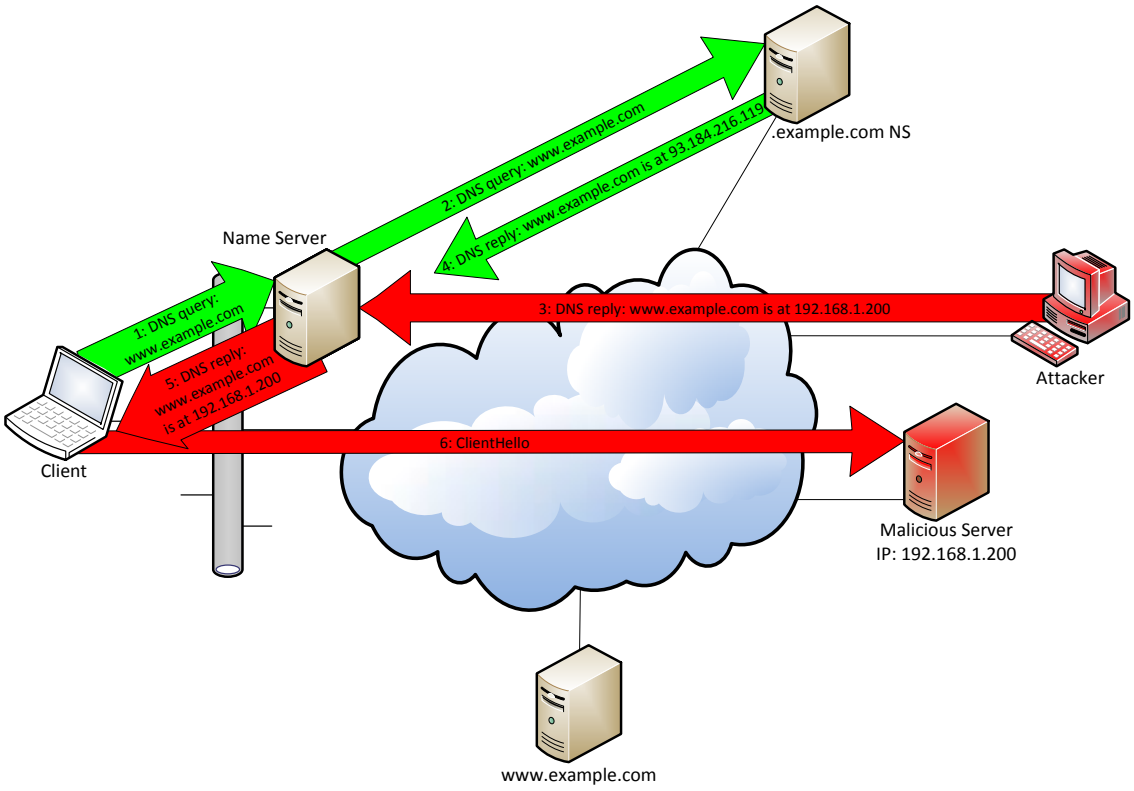

Figure 8. The DNS Cache Poisoning attack.

In Figure 8, the client's resolver normally sends a recursive DNS query to its domain's resolving name server which then iteratively begins to solve the IP address by sending the query (in this case) to the .example.com-domain's name server. The .example.com-domain's name server normally responds by sending a DNS reply, where it informs, that the target web server is located at the 93.184.216.119, but the reply is dropped by the resolving name server because the reply, sent by the attacker, is received first. The resolving name server will accept the first reply with the correct transaction identifier (TXID) and the correct User Datagram Protocol (UDP) source port for a DNS query message. The resolving name server will store this malicious resource record to its cache for a pre-set TTL-value. This incorrect information is also transmitted to the client who then with no doubt believes that the www.example.com is located at 192.168.1.200 instead of 93.184.216.119 and begins to negotiate the connection with the malicious web server. [2; 3; 4]

DNS is a well-functioning and dynamic system but unfortunately it is also very vulnerable to different malicious actions. Its internal functionality does not provide proper protection against these attacks and the framework to build one is insufficient.

2.3 Domain Name System Security Extensions

The Domain Name System Security Extensions (DNSSEC) expand the vulnerable DNS by providing a data origin authentication and data integrity by associating cryptographically generated digital signatures. By using DNSSEC, a DNS client can be sure that the received DNS reply and its contained information about the target location's IP address are reliable. This will ensure that the IP address belongs to the particularly required domain name. DNSSEC does not provide data confidentiality. [5]

DNSSEC's first specification, which was introduced in the IETF RFC 2535 standard in March 1999, has become obsolete by newer IETF RFC 4033, RFC 4034 and RFC 4035 standards in March 2005. At first, in this chapter, the functionality of the DNSSEC is briefly presented.

2.3.1 General

DNSSEC is a set of extensions, which offer a data origin authentication, data integrity and an authenticated denial of data existence. In order to provide those, DNSSEC required some changes to the DNS protocol. Four new resource record (RR) types were added: two new message header bits, a support for larger DNS messages sizes and a support for querying DNSSEC data in response messages. [6]

The first of four new RR types is the DNS Public Key (DNSKEY) RR. DNSKEY RR contains the public keys, which are used in the DNSSEC authentication process. The second RR type is the resource record digital signature (RRSIG), which stores the digital signatures used in the DNSSEC authentication chain. The third RR type is the Next Secure (NSEC) and its latest version NSEC3, which authenticates denial of existence. The fourth RR type is the delegation signer (DS), which provides hierarchical authentication between the child zone and the parent zone. [7]

Two new message header bits are Checking Disabled (CD) bit and Authenticated Data (AD) bit. These bits are used for communication between security-aware resolvers and security-aware recursive name servers. The CD bit allows a security-aware resolver to disable signature validation. The AD bit is set by the security-aware name server if and only if the RRs in the response are authentic. [8]

By setting the DNSSEC OK (DO) bit on the resolver informs the server that it is able to accept DNSSEC RRs [9]. The DO bit is included to the Extension Mechanisms for DNS (EDNS) which allows, for example, larger message size for the DNS packet that DNSSEC requires [10].

2.3.2 Functionality

The DNSSEC is an extension for the DNS, thus the basic structure and functionality are the same. The client's DNS resolver first sends the DNS query message to the resolving

name server. The difference compared to the standard DNS process is that the DO bit is set and the client's resolver accepts and can handle DNSSEC replies. The beginning of the DNS process with the DNSSEC support is shown in Figure 9.

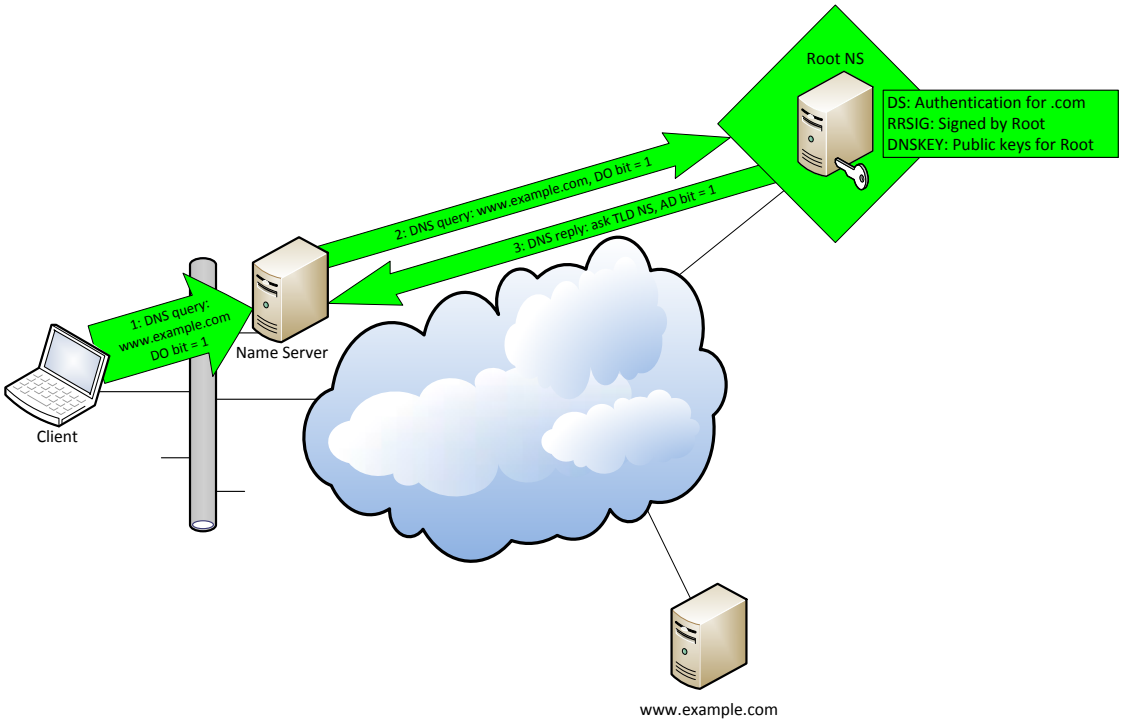

Figure 9. The DNS query to the root with DNSSEC.

If the recursive and resolving name server does not have DNS entries related to the queried www.example.com entry, it begins the query from the root server and also sets the DO bit to 1. Since the DO bit is a part of and defined by the EDNS all the devices must support it. The root server has the information about the .com TLD's name server and it replies. It also detects the set DO bit so it includes the various DNSSEC resource records into the DNS response and sets the AD bit to 1.

There are two different types of keys the root has and the public ones are stored in the DNSKEY RR. The Zone Signing Key (ZSK) is for signing the zones and the Key Signing Key (KSK) for signing the keys. The public keys of the root are verified by the Internet Corporation for Assigned Names and Numbers (ICANN) and well-known Certificate Authority (CA), VeriSign, Inc., and the root has signed its own certificate by using its own private ZSK since July, 2010. This signature is stored in the RRSIG.

The root works as a parent zone to the TLD .com child zone. The root has signed the KSK for the TLD .com by using the root's ZSK and stored this information to the DS RR. Thus, the root and the TLD .com have created a chain of trust between each other. Because of this, any DNSSEC zone, which completes the chain of trust from the root, can be validated by using the root trust anchor. [7]

When the response from the root is received, the resolving name server sends a query to the TLD name server as can be seen in Figure 10.

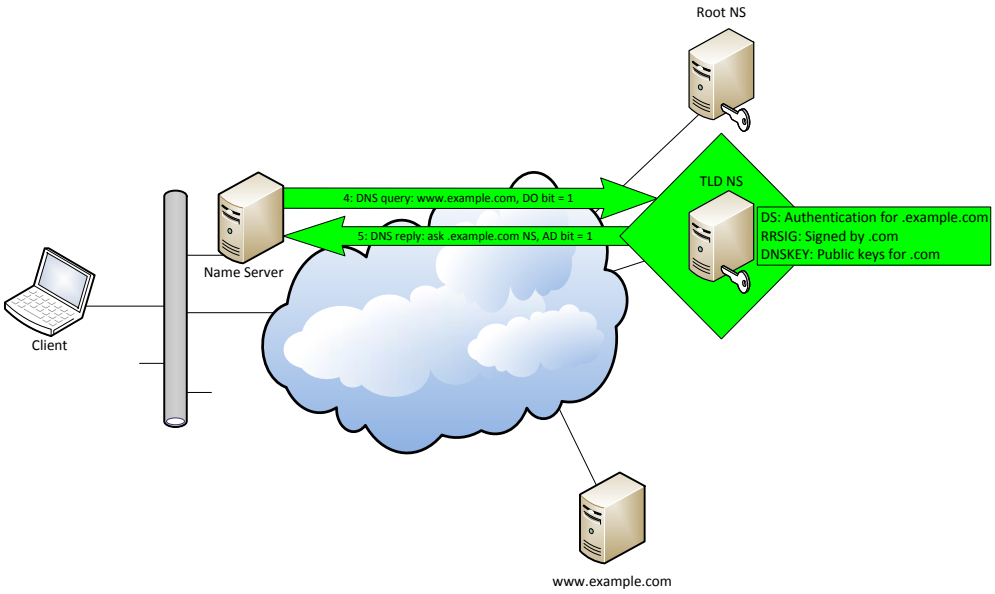

Figure 10. The DNS query to TLD NS with DNSSEC.

The query is the same and the DO bit is set. The TLD .com name server does not have information about www.example.com server, but it has information about .example.com name server's location. The TLD .com name server responds to the resolving name server and because the DNSSEC is admitted, it adds the DNSSEC RRs to the response message and sets the AD bit to 1.

The DNSKEY record contains the public ZSK and KSK keys for the .com zone. These are the same keys that the root zone has signed and stored in its own DS record field seen in Figure 9. The RRSIG record contains .com zone's own signature. The DS contains authenticated public keys for the .example.com zone which the .com zone has signed by using its KSK. [7]

When the response from the TLD is received, the resolving name server then queries the name server of the .example.com zone as shown in Figure 11.

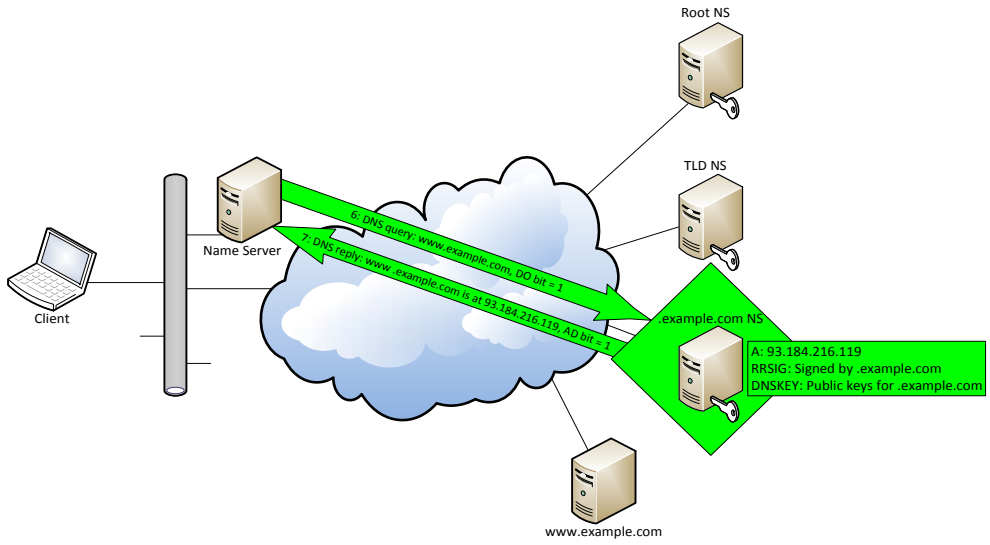

Figure 11. The DNS query to the .example.com name server with DNSSEC.

As requested, the .example.com name server also responds with the DNSSEC information instead of just the normal DNS reply. With these DNSSEC records, .example.com name server informs the resolving name server. The public ZSK and KSK keys, which .com zone authenticated, are stored into the DNSKEY record, the RRSIG record is signed by the .example.com zone's own signature and a normal A record, which contains the location (that is IP address) of the www.example.com server. All the records are authenticated so the AD bit is still set. [7]

The resolving name server then forwards this received information to the client's resolver, which received the response to its original query and now has the IP address of the queried web server of the .example.com. Now the client's web browser can begin to negotiate the connection between the client and the web server. This process is shown in Figure 12.

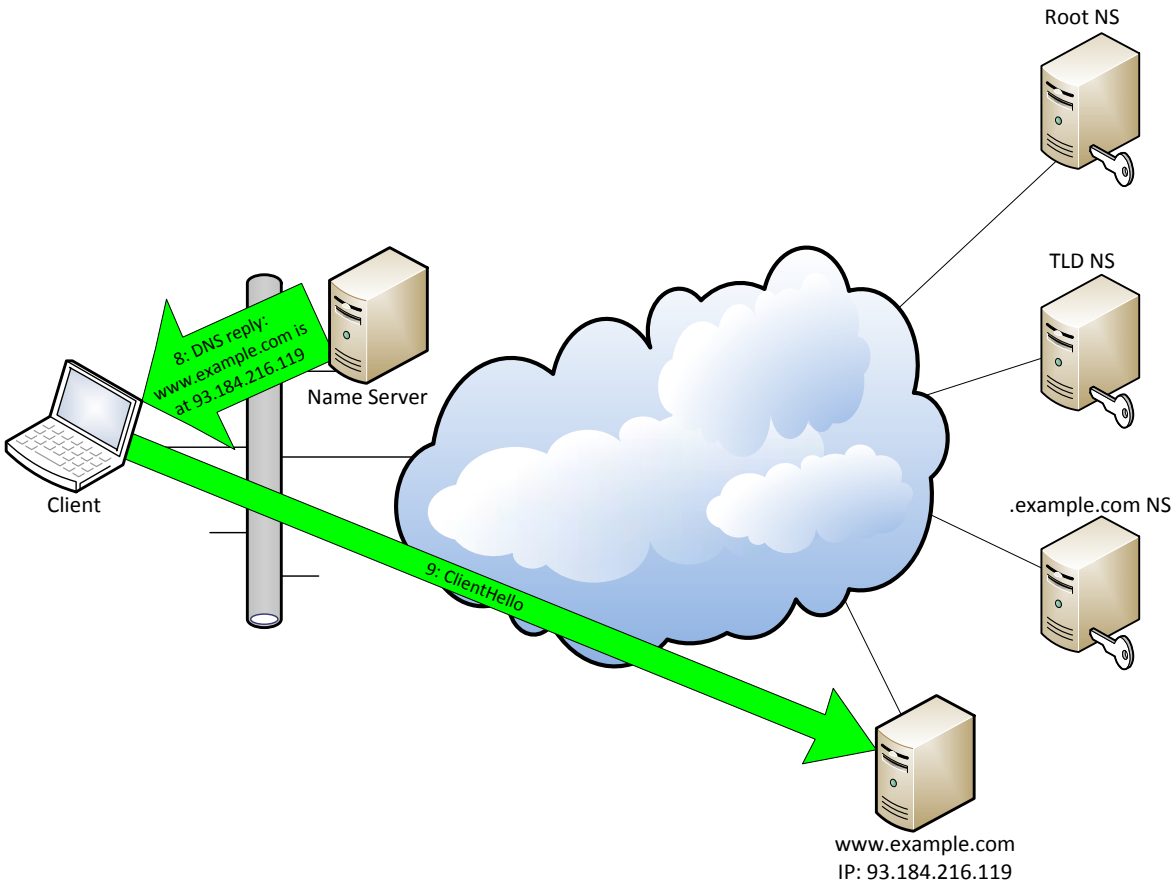

Figure 12. The final step on the path of the DNS query with DNSSEC.

The DNS query process with the DNSSEC has finished and, for example, the TLS handshake process has begun.

2.3.3 Chain of Trust

The idea of the DNSSEC is to provide a global system, which offers reliable DNS entries. The DS RR provides an authentication chain between the trusted root and the destination web server. The term "chain of trust" is based on the structure, where, at first, the root is publicly well-known and verified. Then the root authenticates its child TLD zone, for example, .com zone, which then authenticates its child zone (in this case, .example.com zone). The .example.com name server is now authenticated and knows the IP address of the web server, that is, it responds with the A record, so the resolving name server and the client's resolver can be sure, that the IP address belongs to the correct, queried domain name. [8]

A parent can delegate an authentication to the child. When the child domain name is already registered under a TLD there is one authentication mechanism to use to become secure. For example, if the child .example.com zone is upgrading its DNS to DNSSEC and its parent zone, .com, is already using DNSSEC, the following tasks have to be taken.

1. The .example.com generates a public-private key pair.
2. The .example.com signs its own zone.
3. The .example.com informs the .com parent zone about its preparedness.
4. The .com checks the IP addresses of the NSs of the .example.com by checking the TLD database.
5. The .com gets the .example.com's key by using DNS.
6. The .com traceroutes the .example.com.
7. Depending on the validations, the .com either signs or drops the key and informs the registrant and the domain holder.
8. The .com inserts the signed DS RR to its own zone.
9. The .example.com receives an acknowledgement response from the .com.

It is very crucial at these early steps to keep the possible attackers out and prevent them from getting involved in any cases. The worst result may be that the attacker rules the whole zone. [21]

It is at the security-aware name server's responsibility to verify DNSSEC signatures. It trusts to the authenticated root's public KSK key. The authentication of the root zone is made by the root itself by using the private half of the public-private ZSK key pair. The root's public ZSK key has been signed by the private half of the public-private KSK key pair. The authentication of the root zone can be made by using the public half of the public-private ZSK key pair and the authentication of the ZSK public key can be

made by using the public half of the public-private KSK key pair, which is already trusted.

The beginning of the chain of trust is shown in Figure 13, where ICANN has verified the root's public KSK key, which private half has signed the root's public ZSK key, which private half has signed the root zone. This root zone has information of the .com zone. [5; 7; 11]

In Figure 13, by the syntax KSK_X^+ is meant, that the public KSK is managed by X and by the syntax KSK_X^- is meant, that the private KSK is managed by X. The same syntax applies for the ZSKs.

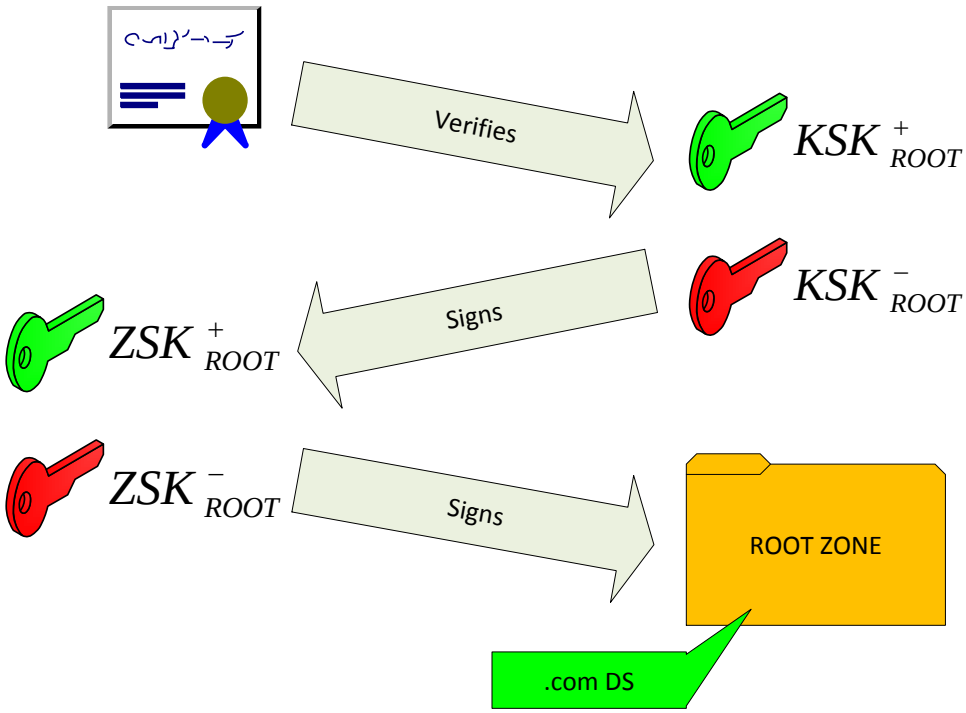

Figure 13. The authentication method of the root zone.

The DS RR is the authentication chain link between the DNS zone boundaries. The DS RR field contains the digest of the child zone's DNSKEY RR field, it refers to. The referred DNSKEY RR is, in this case, the .com zone's DNSKEY RR. The digest is a hash of the DNSKEY owner name and the data, which contains the flag, the protocol, the algorithm and the public key. [7]

Next, the security-aware name server receives the response from the TLD .com zone's name server. It already has received the DS RR for this zone so it has the hash of the public KSK key for this particular zone and by using this information the security-aware name server can verify the ZSK and so on the whole .com zone. This process is shown in Figure 14. [7]

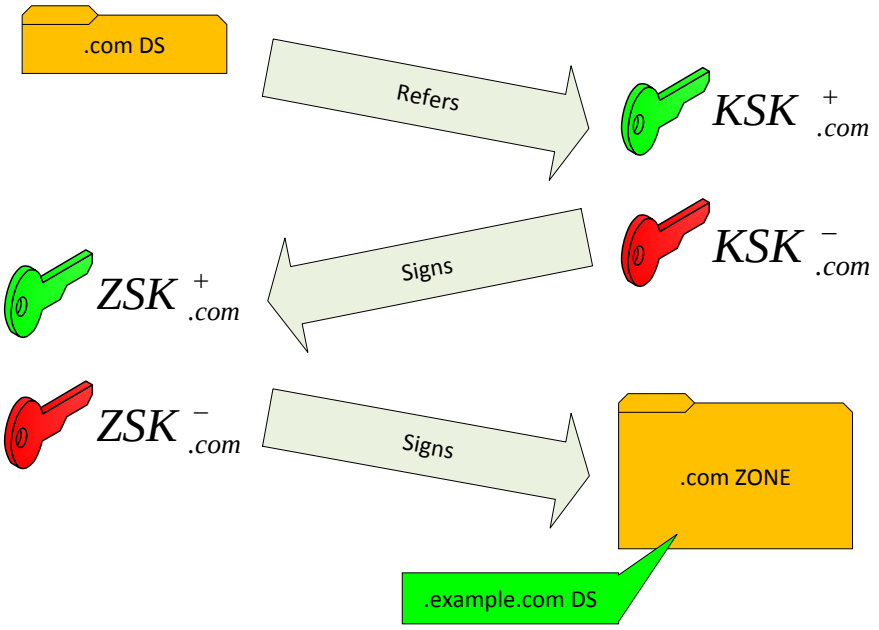

Figure 14. The authentication method of the .com zone.

As in the response from the root, also in the response from the .com name server there is the DS RR field, which now contains the DNSKEY RR field digest of the .com zone's child zone, which, in this case, is the .example.com zone.

The verified DS RR now authenticates the following DNS response message's DNSSEC records received next from the .example.com zone's name server and shown in Figure 15.

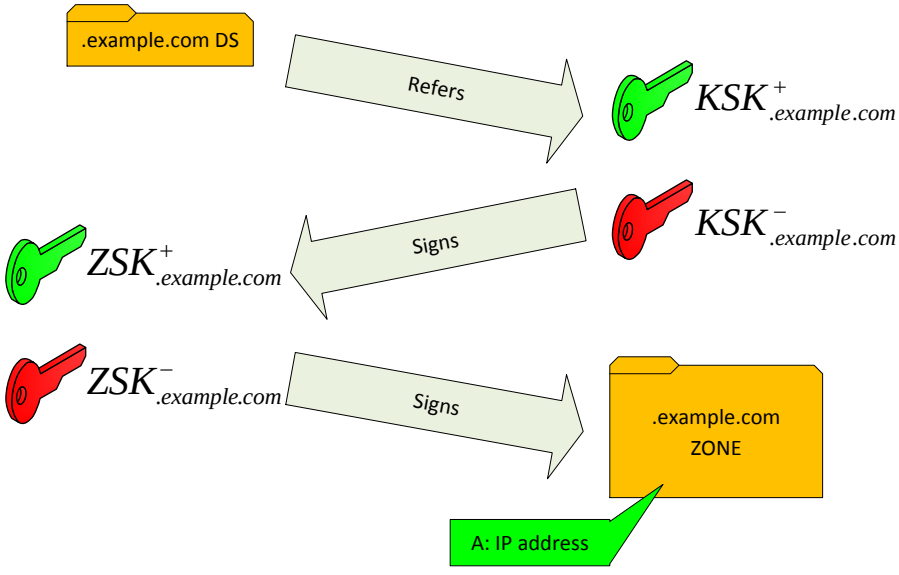

Figure 15. The authentication method of the .example.com zone.

By using the public KSK key, the security-aware name server can verify the .example.com zone and also the A record, which contains the IP address of the queried domain name. [7]

The chain of trust for this particular case is shown in Figure 16.

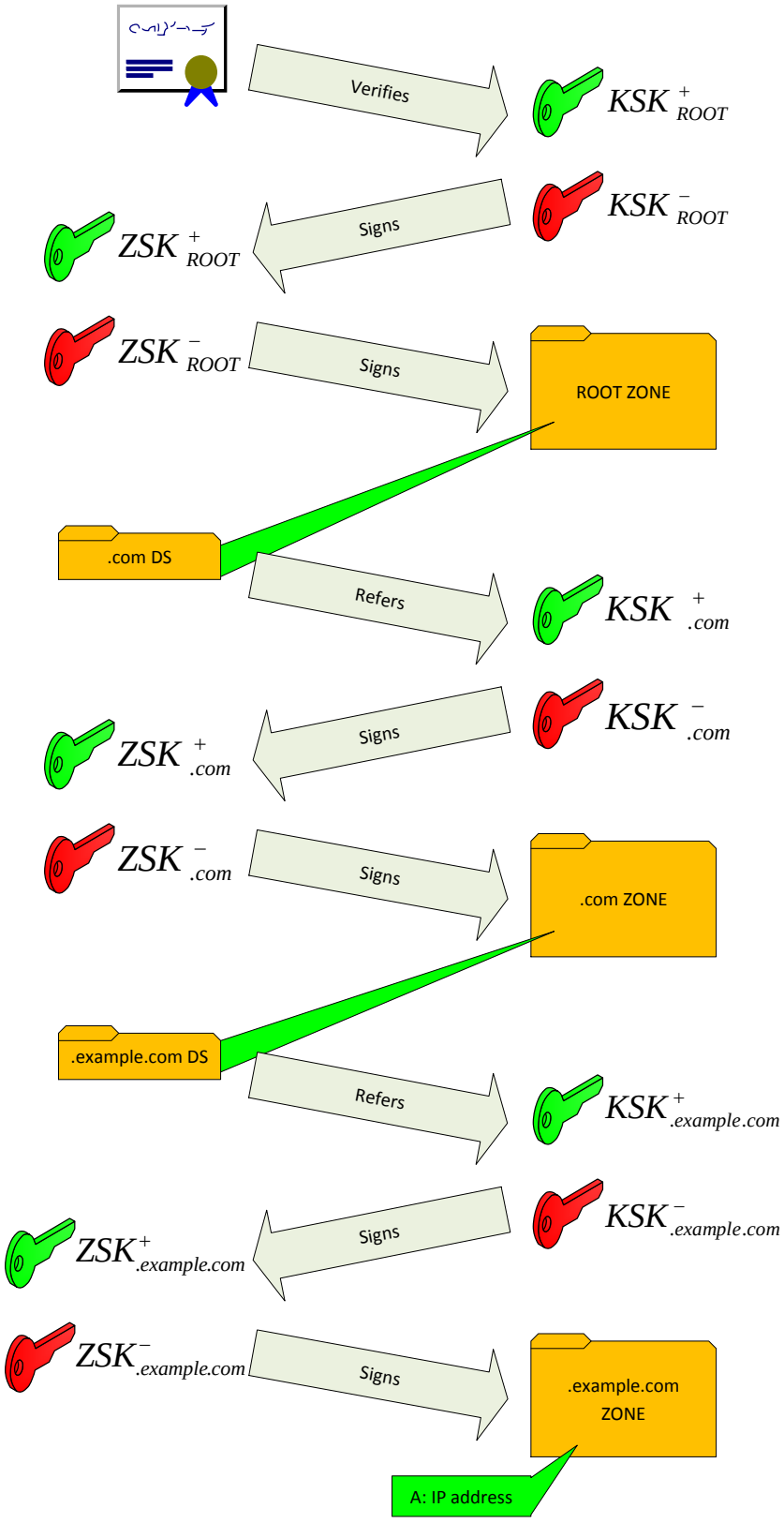

Figure 16. The chain of trust for this case.

The chain of trust lies on the globally trustworthy root, which is the topmost parent or it lies on a Trust Anchor, which is obtained to the validating security-aware name server

via methods outside the DNS protocol. The Trust Anchor is a configured DNSKEY RR or a DS RR, which the validating security-aware name server will use, instead of the root, as a starting point for building the chain of trust. [5]

2.3.4 Challenges of DNSSEC

Improved security increases the complexity significantly, which leads to the situation, where increased baseline expertise is also required in every zone using DNSSEC. The implementation of DNSSEC to the global, existing and continuously expanding Internet and its hierarchical DNS produced in different ways needs a standard and methods which are highly compatible. Many operators and organizations have to update their DNS software and might have to upgrade their DNS hardware. The delays and the response times increase during DNS lookups because of the extra processing times needed for the key and signature verifications. In Figure 17, the performance of DNSSEC compared to the queries without it is presented.

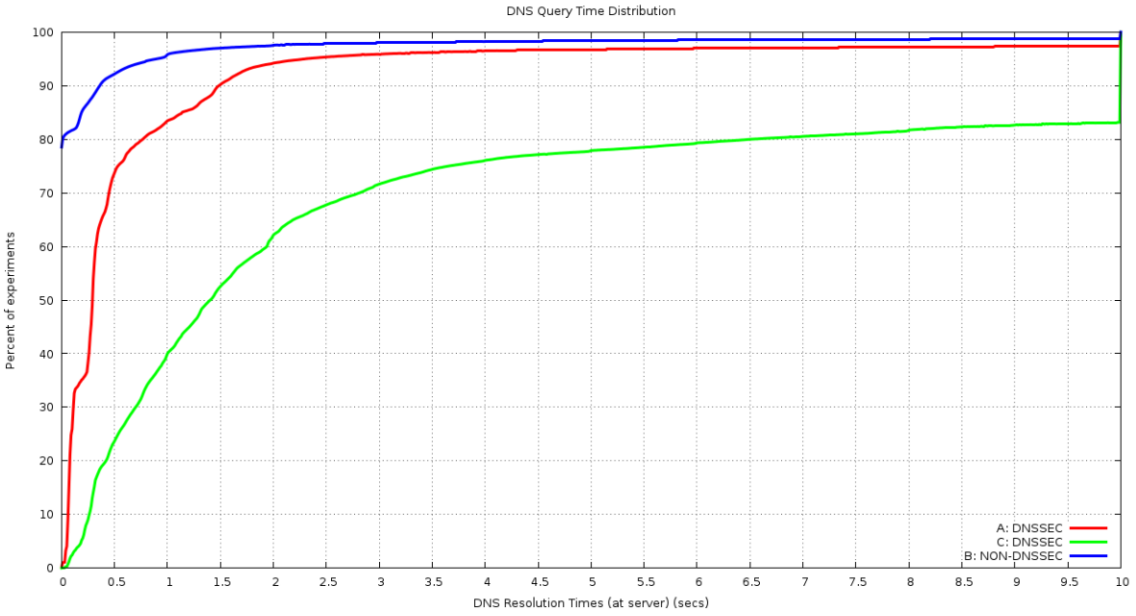

Figure 17. The performance of DNSSEC [17, p. 19].

In Figure 17, the ”A” (red line) corresponds to the queries with the DNSSEC-validating resolver, the “B” (blue line) corresponds to the queries with the non-DNSSEC-validating resolver and the “C” (green line) corresponds to the queries with the DNSSEC-validating resolver, but the bad signature leads to the failure of the validation. It is shown in Figure 17, that almost 20 percent of all the queries that use the DNSSEC have a response time longer than one second, whereas in queries, which do not use the DNSSEC, the corresponding percent is less than 5 percent. It is also shown, that the bad signature causes more than 10 second response times to more than 15 percent of the queries. Overall, the DNSSEC might cause a slower user experience to most of the end users even though the whole DNS process should be invisible to them. [12; 16]

One of the biggest challenges is the key management, which includes the key storing, key generation and key replacement, called rollover. The KSK is used to sign a key set, which has a minor operational effect, because it signs only a little section of the zone data. The KSK can be made stronger and that is why it can be updated much less frequently than other keys. The KSK is the long-term key, which is usually changed once in a year or two. It is crucial, that the private half of the KSK public-private key pair is stored offline and in a safe location. [13; 15]

The ZSK is a short-term key and is changed more frequently, usually once in a month or two. Differentiating the KSK and the ZSK, allows the zone administrator to update the ZSK and to re-sign the zone data by using this new ZSK, re-signed with the KSK, without negotiating with the parent zone. [13; 15]

The keys need to be changed periodically, because the probability of a key getting compromised because of carelessness, accident, espionage or cryptanalysis increases the longer the key is in use. The more frequently the keys are changed, the more difficult it is for the attacker to perform its malicious actions. The rollover process for the KSK needs to be done in cooperation with the parent DNSSEC zone. The process begins with the initial state where the parental DS RR points to the child's current DNSKEY. Time takes a crucial part in the DNSSEC rollovers, so the child has to be accurate with the TTL values of that DS RR mentioned above.

The next step is called the "new DNSKEY" phase, where the zone administrator generates a new KSK. When this new key is delivered to the parent, the parent will generate and publish a new DS RR, which points to the new DNSKEY. When the TTL expires the parent replaces the old DS RR with the new DS RR. The child removes the old DNSKEY. This process is based on the Double-Signature Key Signing Key Rollover. There are a few rollover variants. In a Double-DS Key Signing Key Rollover, the parent has two different DS RRs at the same time, which are pointing to the two different DNSKEY respectively until the TTL expires. In a Straightforward Rollover in a Single-Type Signing Scheme, a new DNSKEY is presented and all the RRsets are signed with the old DNSKEY and the new DNSKEY until the TTL expires. In a Double-DS Rollover in a Single-Type Signing Scheme, a new DNSKEY and a new DS RR are provided to the parent. The new DNSKEY is not yet used to sign the RRsets until the TTL expires. [14; 15]

There are two different methods to ensure that during the rollover for the ZSK, the zone can still be verified. The first method is called the Pre-Publish Zone Signing Key Rollover and the second method is called the Double-Signature Zone Signing Key Rollover.

The Pre-Publish Zone Signing Key Rollover has four steps. The First step is called the Initial step, which presents the current state. The next step is called the "new DNSKEY" step, where the new ZSK is introduced but it has not been used yet to generate the new signatures. The next step, in which the data in the zone is signed, is called the "new RRSIGs" step. The new ZSK is used to generate the signatures in the zone and all the old signatures generated by the old ZSK are removed from the zone. The old

ZSK still remains valid, so the cached data can still be verified until the TTL expires. The last step is called the "DNSKEY removal" step, where the old ZSK is removed and only the new ZSK remains. The whole key set is re-signed with the current KSK. [15]

This Pre-Publish Zone Signing Key Rollover method can be simplified by generating the next ZSK and introducing it right after the rollover. The new ZSK remains stored until the next rollover phase. [15]

The second method is called the Double-Signature Zone Signing Key Rollover. There are three steps in this method. The first step is called the Initial step, which presents the current state. The next step is called the "new DNSKEY" step. In this step, the new ZSK is introduced and all the zone data is signed with this new ZSK. There is a phase, when all the zone data is signed with the new ZSK and also with the old ZSK until the TTL expires and the old ZSK and all its signatures will be removed at the last "DNSKEY removal" step. Only the new ZSK remains and the whole key set is re-signed with the current KSK. [15]

2.4 Problems and Needs

DNS is a critical infrastructure of the Internet because it is in use everywhere and because the Internet, as it is known nowadays, would not work without it. DNS has remained almost the same from the beginning, even if many things around it have changed. When DNS was developed, there were none of those threats, which are malicious to DNS and which exist nowadays. DNS has no protection against them. That is, why DNSSEC was developed.

In the DNSSEC system, all the parents can delegate authentication forward to their children to achieve an unbroken chain of trust. In the steps of authentication delegation, there is a great responsibility for the DNS operators and DNSSEC registrants to provide a reliable mechanism to confirm a child's trustworthiness.

DNSSEC provides a sufficient reliability and data integrity to DNS. DNS's responses to the client's resolver are verified and valid. The client's web browser begins to negotiate a TLS connection between a bank's web server and itself. It has a verified IP address where to send a "clientHello" message. After a while, the client receives a response from the server, which includes a server's certificate. By verifying this certificate, the server is authenticated. The verifying is done by the third party CA. The browser trusts by default a certain amount of CAs, which are already configured to the system. Any of these CAs could verify any other domain. That means the weakest CA defines the reliability. Unfortunately, there have been the breaches, for example in March 2011, a malicious attacker managed to obtain digital certificates including Google, Yahoo and Skype from the CA Comodo [19] and in August 2011, an another breach in DigiNotar, which issued a digital certificate for google.com to someone, who is not Google [20]. These breaches allowed attackers to use what is known as a man-in-

the-middle attack to monitor any traffic including passwords and emails, even if the traffic was protected with an encrypted TLS connection [19].

As it is shown in Figure 18, when the client receives a valid and DNSSEC verified response from the NS, there is still a possibility for an attacker to perform a man-in-the-middle attack to the TLS connection.

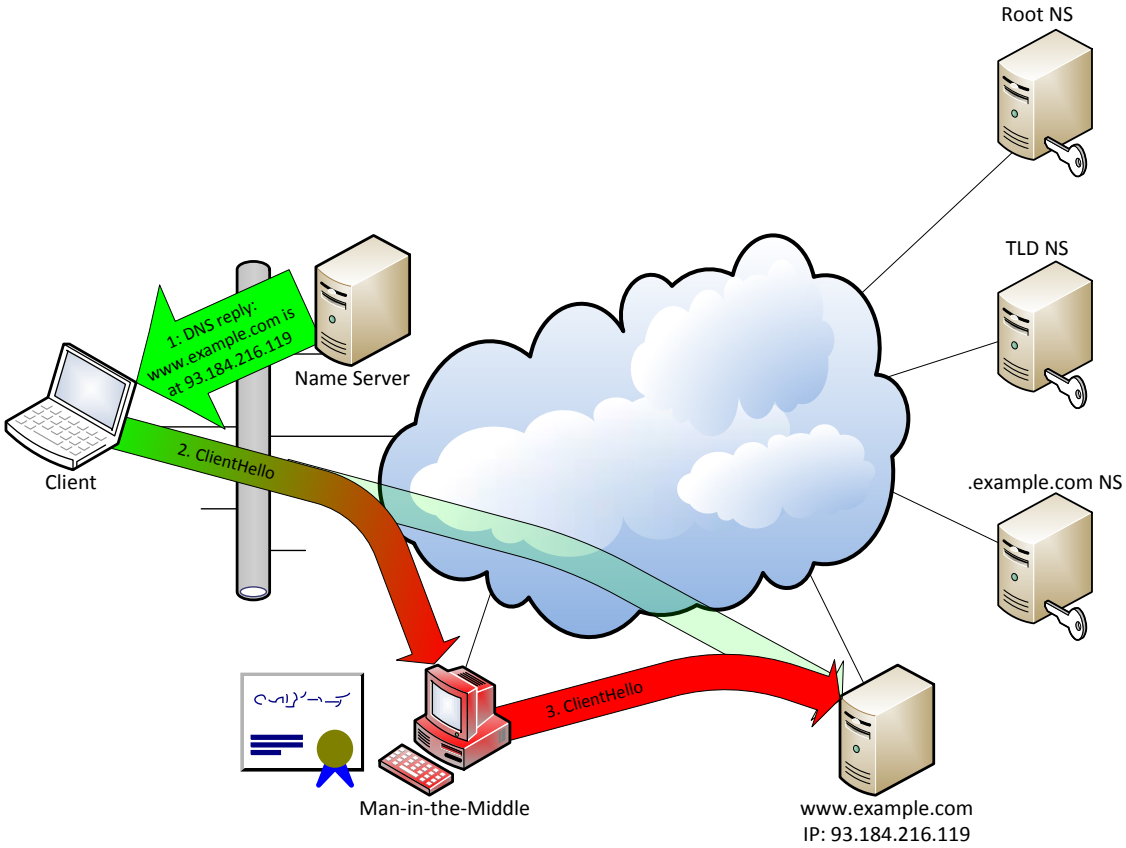

Figure 18. Man-in-the-Middle attack.

The malicious web server has a valid certificate issued by a weak CA and because the client's web browser trusts the weak CA, the client believes to communicate with the correct, trustworthy web server. The attacker can now eavesdrop all the client's passwords and other personal data. There is clearly a need for a relation between the DNSSEC records and the TLS certificate. It assures that a client uses the predetermined certificate.

DNSSEC does not protect against human made errors. If the client is attempting to connect to the web server locating at www.example.com and he/she, by misspelling the domain name, enters to the web site located at www.example.org, which currently could be under the attacker's control. The fake web site could look exactly the same as the original one and nothing can prevent the client to reveal his/her secrets to the attacker.

3 DANE

This chapter introduces the theoretical and technical perspective to the main subject, the DNS-based Authentication of Named Entities (DANE). The general information, full functionality, the protocol structure and the different applications of the DANE are presented in detail.

3.1 Utilization

The DANE protocol was developed to improve the TLS authentication by allowing the domain name administrators to bind their used certificates to the DNS names using the DNSSEC and a new resource record type [25]. By doing this, the domain name administrator defines which particular certificate issued by the CA is valid for the TLS connection. A security-aware client with DANE support can see the “lock” icon, which signifies that the connection can is encrypted, but he/she can also be sure that the TLS certificate, which is used to encrypt the connection, is the one the domain administrator wants it to be. The DANE protocol specification brings also the opportunity to use the domain administrator's own self-signed certificates without using the third party CA. [30; 31]

This protocol specification uses the TLSA resource records (TLSA RRs), which are received by using the DNS query and validated by using the DNSSEC, to verify a secure TLS negotiation. The DNSSEC validation state [5] must be secure, so that the TLSA RR can be used and the secure TLS handshake can begin. The received TLSA RR, which includes information about the used certificate, is compared to the certificate, received from the TLS server during the TLS handshake. [22; 29]

The DANE-aware browser, or the browser add-on application, can warn the client if the certificate sent by the TLS server does not match to the TLSA record verified by DNSSEC. The connection under negotiation may become insecure. The successful DANE validation is shown in Figure 19.

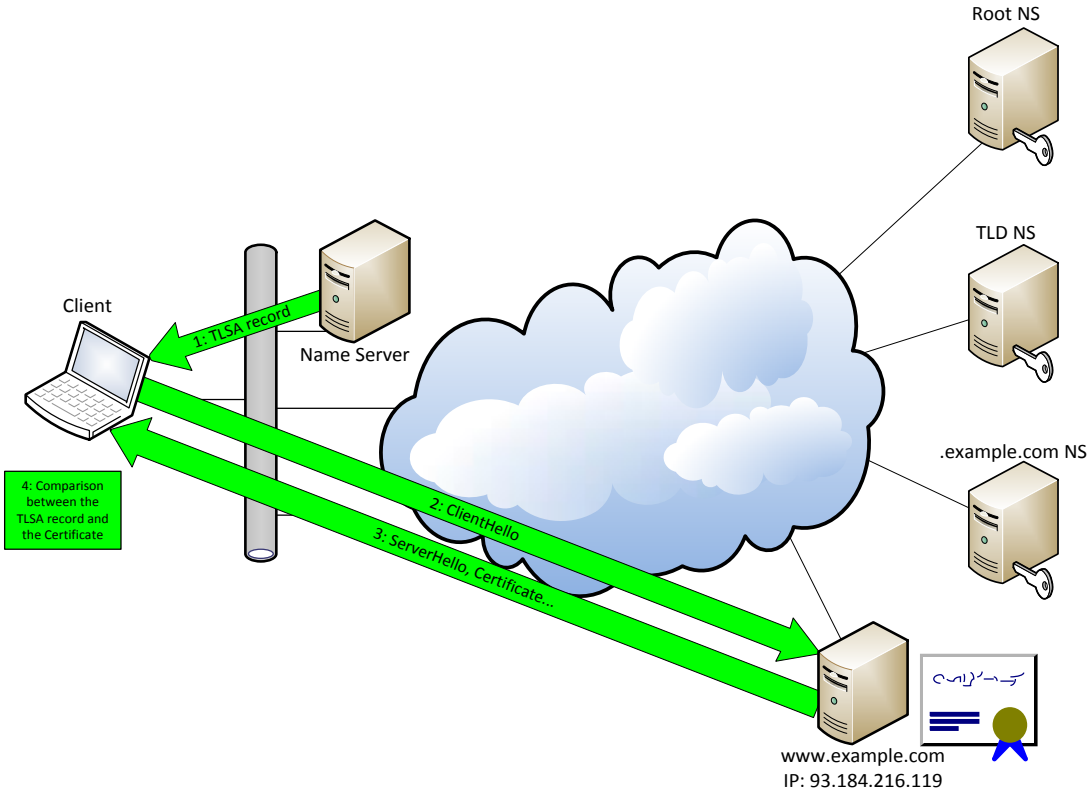

Figure 19. TLS handshake with TLSA record.

The DANE working group has introduced DANE in the IETF RFC 6698 standard in August 2012. Further applications, such as DANE-compatibility with, for example, the email has already been considered and the IETF Internet-Drafts has been developed.

3.2 TLSA

The association between the domain name and the TLS server certificate or the public key is created by using the TLSA RR, which includes certificate usage field, selector field matching type field one-octet each and the certificate association data field. The TLSA resource record data (RDATA) is shown in Figure 20.

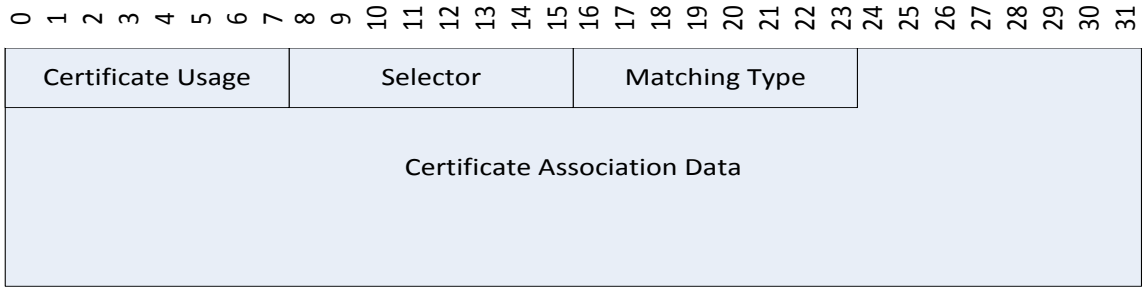

Figure 20. RDATA for TLSA [22].

3.2.1 Certificate Usage Field

The value of the certificate usage field defines how to associate the introduced certificate during the TLS handshake and must be represented as an 8-bit unsigned integer. The values from zero to three are defined. The values from 4 to 254 are unassigned and the value 255 is for the private use. This is shown in Table 1. [22]

Table 1. TLSA certificate usages and proposed acronyms [26, p. 3].

Value	Acronym	Description
0	PKIX-TA	CA constraint
1	PKIX-EE	Service certificate constraint
2	DANE-TA	Trust anchor assertion
3	DANE-EE	Domain-issued certificate
4 – 254		Unassigned
255	PrivCert	Reserved for Private Use

The value 0 is used for a CA constraint certificate usage. This means, that the presented certificate, by the server in the TLS, must pass the validation and a certificate from the TLSA record must be a part of that exact validation process. This certificate usage limits the number of the CAs issuing the certificates for that particular server. [22]

The value 1 is used for a service certificate constraint usage. This means, that the presented certificate, by the server in the TLS, must match the certificate presented in the TLSA record. This limits the number of certificates used for validation. [22]

The value 2 is used for a trust anchor assertion, which allows the administrator of the domain name to define a new trust anchor, which is presented in the TLSA record. The certificate, given by the TLS server, must pass the validation process using this new trust anchor. [22]

The value 3 is used for a domain-issued certificate usage, which allows the administrator of the domain name to present its own certificates without using a third party CA. The presented certificate, by the TLS server, and the certificate in the TLSA record must match. [22]

3.2.2 Selector Field

The value of the selector field defines how the certificate will be matched and must be represented as an 8-bit unsigned integer. The values from zero to one are defined. The values from 2 to 254 are unassigned and the value 255 is for the private use only. This is shown in Table 2. [22]

Table 2. TLSA selectors and proposed acronyms [26, p. 3].

Value	Acronym	Description
0	Cert	Full certificate
1	SPKI	SubjectPublicKeyInfo
2 – 254		Unassigned
255	PrivSel	Reserved for Private Use

The value 0 is used, when the full certificate binary structure is used to match the received certificate against the associated data in TLSA RR. The value 1 is used, when the Distinguished Encoding Rules (DER) encoded SubjectPublicKeyInfo binary structure is used to match the received certificate against the associated data in TLSA RR. [22; 23]

3.2.3 Matching Type Field

The value of the matching type field defines the format of the associated certificate and must be represented as an 8-bit unsigned integer. The values from zero to two are defined, the values from 3 to 254 are unassigned and the value 255 is for the private use only. This is shown in Table 3. [22]

Table 3. TLSA matching types and proposed acronyms [26, p. 3].

Value	Acronym	Description
0	Full	No hash used
1	SHA2-256	256 bit hash by SHA2
2	SHA2-512	512 bit hash by SHA2
3 – 254		Unassigned
255	PrivMatch	Reserved for Private Use

The value 0 is used, when no hashes are used and the associated certificate is raw data [22]. The value 1 is used, when the associated certificate is hashed by using a SHA-256 algorithm [22; 24]. Either the value 0 or value 1 is mandatory. The hash, made by the SHA-256 algorithm is very compact and is recommended to use.

The optional value 2 is used, when the associated certificate is hashed by using a SHA-512 algorithm [22; 24]. The hash, made by the SHA-512 algorithm is rarely used and commonly reserved for the future use.

3.2.4 Certificate Association Data Field

The value of the certificate association data field defines the value to be compared and must be represented as a string of hexadecimal characters. The format of this data field

depends on the selector and the matching type fields and refers to the certificate given by the TLS server. [22]

3.2.5 TLSA RR Format

The TLSA RR example, where the SHA-512 hash is used to hash the association of a Public Key Infrastructure (X.509) (PKIX) CA certificate:

```
_443._tcp.www.example.com. IN TLSA (
        0  0  2  17a4eab60f01c264f4d30c07851c4041
                 88c8d3120986839b26ce238a3d2fc9ed
                 640bd42d9fb9c9797fea6de814cbb16b
                 f7671ae47575119011442216f64a9608 )
```

Another TLSA RR example, where the SHA-256 hash is used to hash the SubjectPublicKeyInfo association by using the domain administrator's own certificate:

```
_443._tcp.www.example.com. IN TLSA (
        3  1  1 bc8397af9b208bc64efb756938ae5766
                 86ee66321265366b3a441ee42acaf6ab )
```

The first example again by using the proposed acronyms [26], which are easier to remember and more informative than the numeric values:

```
_443._tcp.www.example.com. IN TLSA (
        PKIX-TA  Cert  SHA2-512
                 17a4eab60f01c264f4d30c07851c4041
                 88c8d3120986839b26ce238a3d2fc9ed
                 640bd42d9fb9c9797fea6de814cbb16b
                 f7671ae47575119011442216f64a9608 )
```

The domain name prefix contains the port number on which the TLS service exists and the transport protocol name, which is used in the communication. The port number is the left-most label indicated with the underscore character and the protocol name is the second left-most label also indicated with the underscore character. The domain name is the TLS server's fully qualified DNS domain name. The "IN" indicates the Internet class value. TLSA is the RRtype. The hash is added to the Certificate Association Data field. [22]

3.2.6 Record Rollover

Suppose the .example.com domain zone has one TLSA record for a TLS service on the Transmission Control Protocol (TCP) port 443:

_443._tcp.www.example.com. IN TLSA (
0 0 2 17a4eab60f01c264f4d30c07851c4041
88c8d3120986839b26ce238a3d2fc9ed
640bd42d9fb9c9797fea6de814cbb16b
f7671ae47575119011442216f64a9608)

The rollover process is very similar with the rollover process for a DNSSEC ZSKs by using the pre-publish rollover method [15]. Before the rollover process can begin, the new TLSA record must be generated by using the new certificate or SubjectPublicKey-Info and added that new TLSA record alongside the old TLSA record:

_443._tcp.www.example.com. IN TLSA (
0 0 2 17a4eab60f01c264f4d30c07851c4041
88c8d3120986839b26ce238a3d2fc9ed
640bd42d9fb9c9797fea6de814cbb16b
f7671ae47575119011442216f64a9608)

_443._tcp.www.example.com. IN TLSA (
0 0 2 8cb5b2fb8ed98c0866a765ae63f2dbff
8c8cf6dd4c2a59af08fc395bbda85bd3
50649b39fab1d35a99aa0bcafba0341c
54d8efc07711707bb8589fdbc8cfe9ea)

These new records are distributed to the authoritative name servers. When the current certificate for the TLS server expires, it is switched to the new one and when this is done, the old TLSA record is obsolete and can be removed:

_443._tcp.www.example.com. IN TLSA (
0 0 2 8cb5b2fb8ed98c0866a765ae63f2dbff
8c8cf6dd4c2a59af08fc395bbda85bd3
50649b39fab1d35a99aa0bcafba0341c
54d8efc07711707bb8589fdbc8cfe9ea)

The rollover is completed. [22]

3.3 SMIMEA

The Secure/Multipurpose Internet Mail Extensions (S/MIME) latest version is presented in the IETF RFC 5751 –standard in January 2010. The purpose of S/MIME is to provide cryptographic security to the Multipurpose Internet Mail Extensions (MIME) data such as authentication, message integrity, privacy and data encryption [27]. The specifications and the functionalities of MIME and S/MIME are not presented in this thesis.

SMIMEA is a new DNS resource record type, which provides association between the domain name administrator's certificates and S/MIME by using the secured DNS the same way as DANE. This RRtype work is in progress and is proposed in the Internet Draft by IETF. [28]

3.3.1 Utilization

To associate a public key or a certificate with the associated email address the SMIMEA DNS RR is used. The format of the SMIMEA is similar to the TLSA format presented in Figure 20. Also the certificate usage field format, presented in Table 1, the selector field format, presented in Table 2, and the matching type field format, presented in Table 3 have the same semantics as the TLSA record. The hash is added to the Certificate Association Data field. [28]

3.3.2 SMIMEA RR Format

The "alice@example.com" email address is divided to the left-hand side "alice", called the "local-part", and to the right-hand side "example.com", called the "domain". The separator of these sides is the "@" character which is not included in the SMIMEA RR format. [28]

The "local-part" is hashed by using an SHA2-224 algorithm and it becomes the left-most label. The SHA2-224 hash for "alice" is

38b7e5d5651aaf85694a7a7c6d5db1275af86a6df93a36b8a4a2e771.

The second-most label is a string "_smimecert". The third-most label is the "domain" part. The prepared domain name consists of the previous labels separated with the "." character. The completed domain name is

38b7e5d5651aaf85694a7a7c6d5db1275af86a6df93a36b8a4a2e771.
_smimecert.example.com.

The SMIMEA RR is generated in the same way as the TLSA RR. The SMIMEA RR for the "alice@example.com" is

38b7e5d5651aaf85694a7a7c6d5db1275af86a6df93a36b8a4a2e771.
_smimecert.example.com. IN SMIMEA (
0 0 2 8385c57671fc9bc2f157fa9b6e2ad84d
36dc5a35007ed8e55f3aa2ece410cd9f
c1892a087310043c07d807c55634ed7
af31149aa3a9d9dcc06a09db9ecff0d35).

[28]

3.4 DANE for SMTP

Simple Mail Transfer Protocol (SMTP) was developed for email transmission between Message Transfer Agents (MTAs), such as the sender's email server and the receiver's email server. It was first defined in August 1982 in IETF RFC 821 –standard and the latest update was defined in October 2008 in IETF RFC 5321 –standard [32]. The further specifications or functionalities of the SMTP are not presented in this thesis.

DANE TLSA DNS record provides SMTP transport security between MTAs when the location of the destination SMTP server is received via DNS Mail Exchange (MX) records. The usage of DANE TLSA for SMTP security is proposed in the IETF Internet-Draft and the work is in progress. [33]

3.4.1 SMTP without DANE

There are several serious vulnerabilities when an email is sent today. In the normal case, a client sends its email to the MTA, which is the sender's email server. The sender's MTA begins the DNS query and from the received DNS reply it uses the receiver's DNS MX records to locate the destination MTA, which is the receiver's email server. At this point, the sender's MTA sends the email to the destination email server, which then informs the recipient that the email has arrived. [34]

The DNS MX records can be tampered. In that case, the sender's email server supposes, that the destination email server is the correct one, even if it could be the attacker's fake email server. [34]

DNSSEC provides methods to prevent fake DNS MX records. By using DNSSEC, the DNS data is integrated and the sender can be sure, that the location of the destination email server, announced in the DNS MX record, is correct. [34]

The transmission between the sender's MTA and the receiver's MTA is usually still insecure and can be easily eavesdropped. It is possible to use SMTP over the encrypted TLS connection, but it is rarely used. Even if the TLS connection is in use, it still does not prevent the Man-in-the-Middle attacks. Because the server serves both

TLS and non-TLS clients, it uses the SMTP STARTTLS command to begin the negotiation for the shared secret. This command allows the Man-in-the-Middle to downgrade the connection from the encrypted TLS to the plaintext TCP. This method is seen in Figure 21.

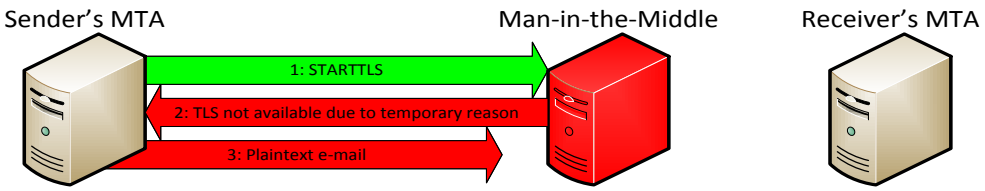

Figure 21. STARTTLS downgrade attack.

The attacker just replies "TLS not available due to temporary reason" to the STARTTLS command and the sender usually continues the transmission without the TLS encryption. The encrypted TLS is dismissed. [33; 34]

3.4.2 SMTP with DANE

The Internet Draft for the SMTP security via opportunistic DANE TLS [33] provides a method for SMTP to prevent a Man-in-the-Middle downgrade attack. This method is based on the DANE TLSA record.

The entire security basis relies on the reliable and hierarchical DNSSEC, which provides authentic MX records. By using these MX records, the client can locate the target server's IP address and can begin to negotiate a TLS connection. When DANE is in use, the TLS connection is enforced. The sender's MTA receives the target server's TLSA record from the target zone's DNSSEC validated name server. This TLSA record is used to verify the target server's certificate in the TLS handshake process. The verification must pass to proceed to an encrypted connection. If the verification fails, the connection is lost and the email is not transmitted.

With DANE, SMTP over TLS is downgrade-resistant and immune to the Man-in-the-Middle attacks. DANE allows, and it is also highly recommended to use, self-signed certificates, which leave the third party CAs off when building the TLS connection. The negotiation of the SMTP TLS with DANE is seen in Figure 22. [33; 34]

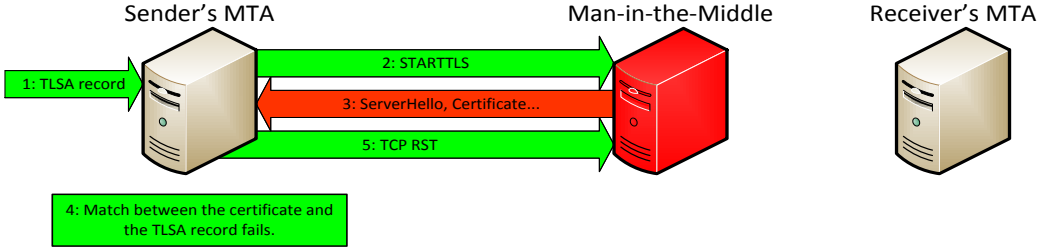

Figure 22. Man-in-the-Middle and the SMTP TLS negotiation with DANE.

As it can be seen in Figure 22, the authentication fails if the attacker offers its own, fake certificate. The TLS negotiation and the whole TCP connection fails if the attacker tries to prevent the encryption and to downgrade the security level from secret to plaintext.

3.4.3 TLSA Record Format for SMTP

The format of the TLSA record for SMTP is similar to the TLSA format presented in Figure 20. The certificate usage field format, presented in Table 1, the selector field format, presented in Table 2, and the matching type field format, presented in Table 3 have the same semantics as the TLSA record but there are a few exceptions in the certificate usage field. The SMTP servers should not publish the TLSA records with the certificate usage 0 or 1. These certificate usages are undefined for the SMTP clients. It is not possible for the SMTP clients to easily maintain a fully completed list of the trusted, third party CAs and these certificate usages do not offer any added security. [33]

The example TLSA record has a SHA-256 hash of the domain administrator's own full certificate

_25._tcp.mail.example.com. IN TLSA (
3 0 1 cc44bab95b9d6e28848cb9bed1c16e73
c7712be73097d9d5851bd553489a2a5d)

The domain name prefix contains the port number on which the TLS service exists. It is 25, when SMTP is in use and the port number is the left-most label indicated with the underscore character. The transport protocol name, which is used in the communication, is TCP, and the protocol name is the second left-most label also indicated with the underscore character. The domain name is the corresponding TLS mail server’s fully qualified DNS domain name. The “IN” indicates the Internet class value. TLSA is the RRtype. The hash is added to the Certificate Association Data field. [33]

4 SMTP WITH DANE IN PRACTICE

This chapter introduces the steps, which were taken on the way to implement a test environment. This platform was needed to test the DANE protocol in practice over the encrypted SMTP traffic.

4.1 Foreknowledge

For this Master of Science Thesis, a small test environment was built, which offered a possibility to perform several tests for DNS, DNSSEC, SMTP, TLS and DANE. The test environment was built by using Oracle's VirtualBox software version 4.3.8. There were no fully complete and ready-to-use test platforms so the environment was built and designed out of the blue. There were a lot of tutorials and configure tips, which were used to build the subsystems, such as DNS, DNSSEC, SMTP and its TLS support and finally, with the combination of all of these subsystems, its DANE support.

4.2 Test Environment Basics

The test environment was built by using four different virtual PCs. Two of these PCs (PC A and PC B) were using the Ubuntu Server 13.10 (Saucy Salamander) operating system and the other two (PC C and PC D) were using the Ubuntu Server 14.04 Trusty Tahr) operating system. The test environment was built and configured as it is shown in Figure 23.

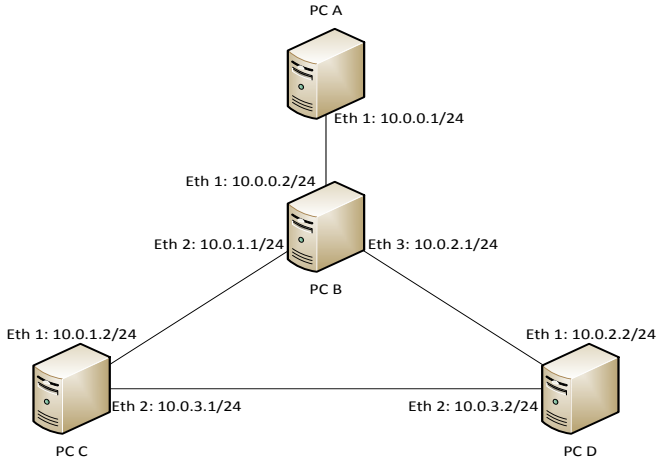

Figure 23. The physical virtual test environment network topology.

All the PCs also had a bridged network adapter, which was connected to the Internet. That bridged network adapter was only used to install all the required software and applications to the virtual computers. That is why it was left out of the physical network topology picture shown in Figure 23. All the other network adapters were VirtualBox Host-Only Ethernet Adapters, which were the new software interfaces created by VirtualBox.

PC A was working as an authoritative name server for the example.com and the example.org zones, PC B was working as a security-aware DNS resolver for the example.com and the example.org zones, PC C was working as a mail server for the example.com zone and PC D was working as a mail server for the example.org zone. PC B was also working as a default gateway for the PC C and PC D, but for the clarity and the packet capturing, a direct connection was built between the mail servers. That connection mirrors the path over the Internet.

After the network adapter configuration, the logical virtual test environment network topology was as it is shown in Figure 24.

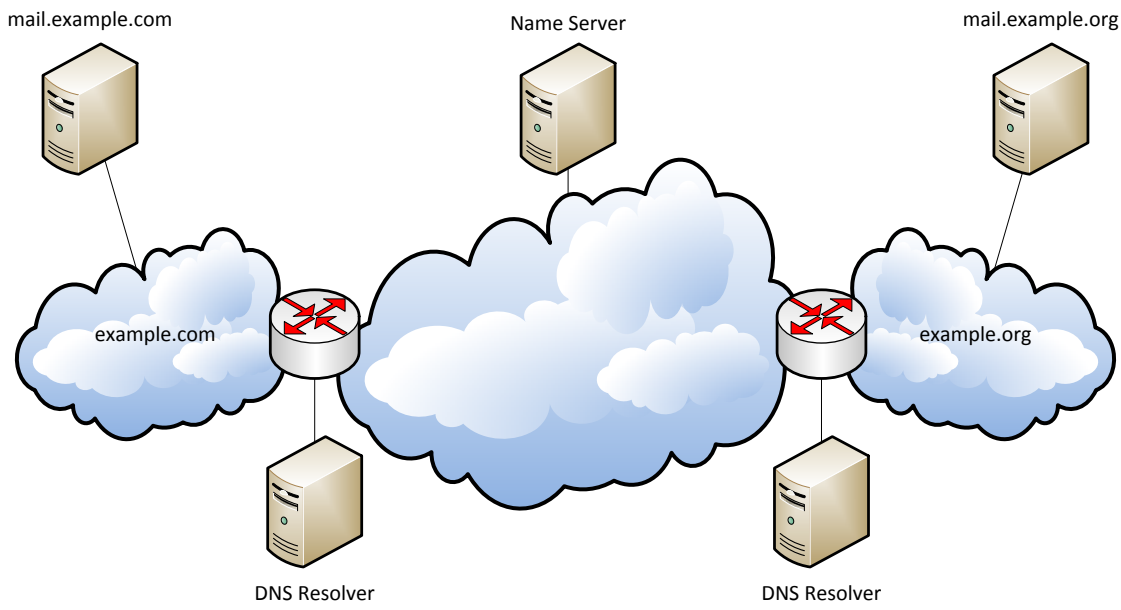

Figure 24. The logical virtual test environment network topology.

As it is seen in Figure 24, the only physical DNS resolver transformed into two different, logical DNS resolvers.

The goal was to transmit encrypted email between PC C and PC D. The encrypted connection was supposed to be implemented by using the SMTP TLS encryption and authenticated by using DANE and its DNSSEC validated TLSA records.

4.3 Configuring DNS

When an email is sent, the sender's MTA wants to know, where the email should be sent, so it begins the DNS query. With this query, it tries to resolve the target mail server's IP address.

The first task was to implement a fully functional DNS for the test environment. The DNS hierarchy was the lowest possible one, as it includes one depth level, one name server. This name server was working as an authoritative server, so it was responsible for the example.com zone and the example.org zone.

4.3.1 Name Server

The software, which was used to provide DNS name service, was Name Server Daemon (NSD) version 3.2.15 implemented by NLnet Labs. NSD was installed to PC A.

The configuration file, *nsd.conf*, was modified so that the NSD server will bind to the IP-address 10.0.0.1 and the port 53. The used zone names were *example.com* and *example.org* and the corresponding zone files were *example.com.zone* and *example.org.zone*. For the reverse lookups, the in-addr.arpa names and zones, related to the mail.example.com and the mail.example.org servers, were also added to the configuration file: the name *1.3.0.10.in-addr.arpa* and the zone file *com.reverse* to the mail.example.com server and the name *2.3.0.10.in-addr.arpa* and the zone file *org.reverse* to the mail.example.org server.

Each of the zone files were created and configured depending on the particular zone and its specifications. In the zone file for the example.com zone, there was an NS record related to the ns.example.com, an MX record related to the mail.example.com and A records for both of them. In the zone file for the example.org zone, there was an NS record related to the ns.example.org, an MX record related to the mail.example.org and A records for both of them. In the zone files for the reverse lookups, there was a pointer record from the corresponding in-addr.arpa name to the host name related to it. [36]

4.3.2 DNS Resolver

Unbound version 1.4.20, developed by NLnet Labs, was used as a proper validating, recursive and caching DNS resolver. The DNSSEC and DANE aware Unbound resolver was installed to the PC B.

The configuration file, *unbound.conf*, was modified to activate server interfaces *10.0.1.1* and *10.0.2.1* to listen to the client DNS queries to the port *53*. Stub zones for the *example.com*, *example.org* and the *10.in-addr.arpa* names were added and the same stub address for all of them was the IP-address of the name server, *10.0.0.1* and its port *53*. [37]

4.4 Configuring DNSSEC

The NSD server and the Unbound resolver were started and the fully functional DNS was running. The next task was to enhance the system by bringing DNSSEC along and signing all the DNS data by using it. The NSD software was already DNSSEC-aware so the main task was to issue proper keys, which were then used for signing the DNS data.

4.4.1 Keys and Signing

The tool *ldns-keygen* was used to generate the ZSK and the KSK key pairs for both of the zones, the example.com zone and the example.org zone. After the generation process, three files, a public key, a private key and a DS file, were generated for each zone and for each key type.

The next step was to use these keys for signing the DNS zone file. The tool, which was used for the signing process, was *ldns-signzone*. The program needs the path of the zone file, the path of the KSK key pair and the path of the ZSK key pair as the input arguments and then, as the output, it generates the signed zone file. The output format is *.signed*, so for the *example.com.zone* file, the signed version would be *example.com.zone.signed*.

4.4.2 Implementation

The location of the zone files had to be changed to the configuration file *nsd.conf* so that they would correspond with the signed zone files. The trust anchors, which in this case were the public KSK keys from each zone, had to be added to the DNS resolver's *unbound.conf* configuration file. After the restart of the NSD server and the DNS resolver the DNSSEC implementation was in action. [36; 37]

4.5 Postfix Mail Services

The program, which was used to provide the necessary email services between PC C and PC D, was Postfix, which was originally developed by Wietse Venema in 1997. Specifically for this test environment, Postfix's latest version 2.11.0 was used because of its DANE support.

4.5.1 Basic Configuration

The Postfix software was installed to PC C and PC D and the basic configurations were set. The mail server configuration type was set to *Internet Site*, which allowed sending and receiving mail directly by using SMTP. The domain for example.com was *example.com*, the corresponding network and netmask was *10.0.1.0/24* and the user was Alice. The corresponding configurations for the example.org domain were *example.org*, *10.0.2.0/24* and *Bob*. Postfix uses mbox as a default mailbox format so it was used in this experiment. [38]

When the Postfix service was started, it was possible to send email from Alice to Bob and vice versa, but the email traffic and packets were plaintext.

4.5.2 Creating a Certificate

The next task was to implement the encryption between the sender's MTA and receiver's MTA by using TLS. Because TLS uses PKI and certificates, the first step, while generating an own self-signed certificate, was to generate a key. The tool *openssl* was used to generate a private key for both of the mail servers. The next step was to generate a certificate signing request for both of the mail servers by using the previously generated key and the *openssl* tool. The generation process requests to enter a Country Name, a State Name, a Locality Name, an Organization Name, an Organizational Unit Name, a Common Name, an Email Address and a few extra attributes. As a result, the generation process produced a certificate signing request, which can be used to create an own self-signed certificate.

To generate an own self-signed certificate, the *openssl* tool was used. It takes the previously generated key and the certificate signing request as an input and generates a certificate as an output. The certificate generations were made to both mail servers. The server certificate file was copied to path

/etc/ssl/certs

and the private key file to path

/etc/ssl/private,

from where any application, when configured, can use them. [39]

4.5.3 Enabling Encryption

When enabling Postfix to provide the TLS encryption, the Postfix's configuration file *main.cf* had to be modified with at least the following configuration additions. The client's SMTP TLS security level was set to the mandatory encryption by adding a

smtp_tls_security_level = encrypt

line to the configuration file. The server's SMTP TLS security level was set to the mandatory encryption by adding a

smtpd_tls_security_level = encrypt

line to the configuration file. This addition means that the server forces clients to use TLS encryption. If the client can not provide encryption, the email is not delivered. That is why this addition should be used only on proper dedicated mail servers.

The locations of the server's certificate and private key had to be added to the configuration file. The path of the certificate file was set by adding a

smtpd_tls_cert_file = /etc/ssl/certs/server.crt

line to the configuration file, informing that the server's certificate was in the file named *server.crt*. The path of the private key file was set by adding a

smtpd_tls_key_file = /etc/ssl/private/server.key

line, also informing that the server's private key file was named *server.key*. After these additions, which were made to both mail servers, the Postfix services were restarted and tested. The test proved that the TLS connection was negotiated before the email was sent. [38; 40]

4.5.4 DANE TLS Authentication

The implemented test platform offered a good basis for increasing security by involving DANE. The Postfix SMTP client supports the opportunistic *dane* level and the mandatory *dane-only* level, which are both based on the DANE TLSA records. The opportunistic *dane* level allows the security level to downgrade itself if TLSA records are not found. That is why the mandatory *dane-only* level was used in this test environment, because it requires the TLSA authentication and there are no fallbacks.

The client's SMTP TLS security level had to be changed from *encrypt* to *dane-only*. This change was made by modifying the line from the file *main.cf* as follows

smtp_tls_security_level = dane-only.

The client's DNS support level had to be set to enable DNSSEC lookups. This was made by adding a

smtp_dns_support_level = dnssec

line to the configuration file. Any DNS MX and address queries from now on request DNSSEC-validated responses. This method required, that a

smtp_host_lookup = dns

line was set and the hosts were able to be found by using DNS. These changes and additions were made to both mail servers, and at the end the Postfix services were restarted. [40]

To provide the DANE TLS authentication, it is also required, that the TLSA record related to the particular mail server's certificate is found from the corresponding DNS zone file. To meet this requirement a

_25._tcp.mail.example.com. IN TLSA (
3 0 1 cc44bab95b9d6e28848cb9bed1c16e73
c7712be73097d9d5851bd553489a2a5d)

line was added to the *example.com.zone* file and a

_25._tcp.mail.example.org. IN TLSA (
3 0 1 f9a52ee9cfa806f7eee68a819ff61e3e
b89fef8ff9b955ec129b3a3ee5a88c33)

line was added to the *example.org.zone* file.

Because the DNS zone files were modified, they had to be signed again by using the *ldns-signzone* tool introduced in Section 4.4.1. After the signing process, the NSD had to be restarted once more. While running again, the fully functional DNS with DNSSEC validation, and a couple of mail servers with SMTP over TLS with DANE authentication, were ready to be tested.

4.6 Testing phase

Before tests with sending and receiving the encrypted emails, responses to the DNS lookups, with DNSSEC and TLSA records, were tested by using a *dig* tool. The

mail.example.org server lookup from the mail.example.com server is shown in Figure 25.

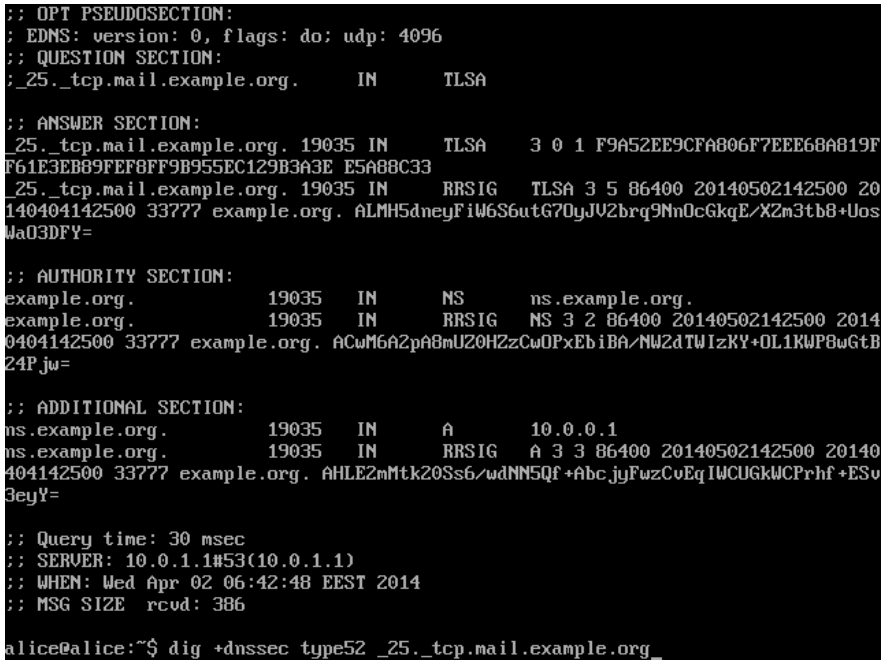

Figure 25. A screenshot of the DNS lookup from mail.example.com to mail.example.org with DNS-SEC and TLSA record.

In the answer section, the TLSA record, related to the TLSA RR introduced in Chapter 4.5.4, and its signature, RRSIG, are seen. DO bit is set. AD bit is also set, but it is not seen in the figure. The mail.example.com server lookup from the mail.example.org server is shown in Figure 26.

Figure 26. A screenshot of the DNS lookup from mail.example.org to mail.example.com with DNS-SEC and TLSA record.

The correct answer section details and the corresponding TLSA RR, related to the value introduced in Chapter 4.5.4, are also seen in Figure 26.

DNS with DNSSEC and TLSA records appeared to be working as they should. The next step was to begin tests with encrypted emails. The Wireshark Network Analyzer program version 1.10.6 was used for analyzing the network traffic and capturing the data packet flows. VirtualBox Host-Only Network interface was set to be the only interface to capture from. With that capture option, Wireshark was aware of all the traffic in the virtual test environment.

For the tests, the caches were emptied. Once the services were running, the tests were begun. The tool, which was used to send an email, was *mail*. With a command

mail bob@example.org

Alice sent an email to Bob, whose mailbox was at mail.example.org server. In Table 4, the DNS queries and responses before transmitting the email are seen.

Table 4. Captured DNS queries and responses from mail.example.com with DNSSEC and TLSA record.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.2	10.0.1.1	DNS	82	Standard query 0x47b3 MX example.org
2	0.012227	10.0.0.2	10.0.0.1	DNS	82	Standard query 0x3977 MX example.org
3	0.012991	10.0.0.1	10.0.0.2	DNS	488	Standard query response 0x3977 MX 10 mail.example.org RRSIG
4	0.020524	10.0.0.2	10.0.0.1	DNS	82	Standard query 0x9f52 DNSKEY example.org
5	0.021901	10.0.0.1	10.0.0.2	DNS	816	Standard query response 0x9f52 DNSKEY DNSKEY RRSIG
6	0.084370	10.0.1.1	10.0.1.2	DNS	488	Standard query response 0x47b3 MX 10 mail.example.org RRSIG
7	0.086053	10.0.1.2	10.0.1.1	DNS	87	Standard query 0x751a A mail.example.org
8	0.091158	10.0.0.2	10.0.0.1	DNS	87	Standard query 0xf8f2 A mail.example.org
9	0.091628	10.0.0.1	10.0.0.2	DNS	388	Standard query response 0xf8f2 A 10.0.3.2 RRSIG
10	0.095188	10.0.1.1	10.0.1.2	DNS	388	Standard query response 0x751a A 10.0.3.2 RRSIG
11	0.101587	10.0.1.2	10.0.1.1	DNS	87	Standard query 0xca0b AAAA mail.example.org
12	0.103837	10.0.0.2	10.0.0.1	DNS	87	Standard query 0x3bc7 AAAA mail.example.org
13	0.104315	10.0.0.1	10.0.0.2	DNS	347	Standard query response 0x3bc7
14	0.108671	10.0.1.1	10.0.1.2	DNS	347	Standard query response 0xca0b
15	0.112271	10.0.1.2	10.0.1.1	DNS	96	Standard query 0x1990 TLSA _25._tcp.mail.example.org
16	0.113822	10.0.0.2	10.0.0.1	DNS	96	Standard query 0x2a79 TLSA _25._tcp.mail.example.org
17	0.114275	10.0.0.1	10.0.0.2	DNS	428	Standard query response 0x2a79 TLSA _25._tcp.mail.example.org RRSIG
18	0.117255	10.0.1.1	10.0.1.2	DNS	428	Standard query response 0x1990 TLSA _25._tcp.mail.example.org RRSIG

There were also queries for the IPv6 addresses, but the AAAA records were not configured to the servers. The answer section of the packet 18, which was a response to the TLSA query, included following descriptions:

_25._tcp.mail.example.org: type TLSA, class IN
Name: _25._tcp.mail.example.org
Type: TLSA (TLSA)
Class: IN (0x0001)
Time to live: 1 day
Data length: 35
Certificate Usage: Domain-issued certificate (3)
Selector: Full certificate (0)
Matching Type: SHA-256 (1)
Certificate Association Data:f9a52ee9cfa806f7eee68a819ff61e3e
b89fef8ff9b955ec129b3a3ee5a88c33

It related to the TLSA RR introduced in Section 4.5.4. The answer section also included the signed TLSA RRSIG, which is described below.

_25._tcp.mail.example.org: type RRSIG, class IN
Name: _25._tcp.mail.example.org
Type: RRSIG (RR signature)
Class: IN (0x0001)
Time to live: 1 day
Data length: 72
Type Covered: 52 (TLSA (TLSA))
Algorithm: DSA (3)
Labels: 5
Original TTL: 86400 (1 day)
Signature Expiration: May 2, 2014 17:25:00
Signature Inception: Apr 4, 2014 17:25:00
Key Tag: 33777
Signer's name: example.org
Signature: 00b307e5d9dec85896e92eaeb46e
cec895766ebabd36739c1a4a84fd
7666ded6fcf94a2c59a3b70c56

When the location of the mail.example.org server was resolved, it was possible to begin with the email transmission. In Table 5, the beginning of the TLS handshake is seen. The TCP three-way handshake has already been negotiated in previous packets, and the connection to the mail.example.org server has been formed.

Table 5. Captured TLS negotiation from mail.example.com to mail.example.org.

No.	Time	Source	Destination	Protocol	Length	Info
32	0.439681	10.0.3.2	10.0.3.1	SMTP	111	S: 220 mail.example.org ESMTP Postfix (Ubuntu)
33	0.442237	10.0.3.1	10.0.3.2	TCP	66	40331 > smtp [ACK] Seq=1 Ack=46 Win=29248 Len=0 TSval=35852082 TSecr=34752520
34	0.443228	10.0.3.1	10.0.3.2	SMTP	89	C: EHLO mail.example.com
35	0.443470	10.0.3.2	10.0.3.1	TCP	66	smtp > 40331 [ACK] Seq=46 Ack=24 Win=28992 Len=0 TSval=34752521 TSecr=35852082
36	0.444588	10.0.3.2	10.0.3.1	SMTP	205	S: 250 mail.example.org | 250 PIPELINING | 250 SIZE 10240000 | 250 VRFY | 250 ETRN | 250 STARTTLS | 250 ENHANCEDSTATUSCODES | 250 8BITMIME | 250 DSN
37	0.445460	10.0.3.1	10.0.3.2	SMTP	76	C: STARTTLS
38	0.446017	10.0.3.2	10.0.3.1	SMTP	96	S: 220 2.0.0 Ready to start TLS
39	0.451906	10.0.3.1	10.0.3.2	TLSv1.2	392	Client Hello
40	0.464818	10.0.3.2	10.0.3.1	TLSv1.2	1497	Server Hello, Certificate, Server Key Exchange, Server Hello Done
41	0.471663	10.0.3.1	10.0.3.2	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
42	0.474789	10.0.3.2	10.0.3.1	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
43	0.477210	10.0.3.1	10.0.3.2	TLSv1.2	118	Application Data
44	0.486165	10.0.3.2	10.0.3.1	TLSv1.2	220	Application Data

When the encrypted email was completely sent, the TLS and TCP connections were closed.

It was confirmed from the mail.example.org server, that the encrypted email from Alice had arrived to Bob. A screenshot is seen in Figure 27.

```
[sudo] password for bob:
 * Starting Postfix Mail Transport Agent postfix                    [ OK ]
bob@bob:~$ mail
"/var/mail/bob": 1 message 1 new
>N   1 alice              Wed Apr  2 01:41  23/738   TLS Works!
? 1
Return-Path: <alice@alice>
X-Original-To: bob@example.org
Delivered-To: bob@example.org
Received: from mail.example.com (mail.example.com [10.0.3.1])
        (using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits))
        (No client certificate requested)
        by mail.example.org (Postfix) with ESMTPS id C457420CFA
        for <bob@example.org>; Wed,  2 Apr 2014 01:41:58 +0300 (EEST)
Received: by mail.example.com (Postfix, from userid 1000)
        id 76F7860838; Wed,  2 Apr 2014 08:37:57 +0300 (EEST)
To: <bob@example.org>
Subject: TLS Works!
X-Mailer: mail (GNU Mailutils 2.99.98)
Message-Id: <20140402053838.76F7860838@mail.example.com>
Date: Wed,  2 Apr 2014 08:37:57 +0300 (EEST)
From: alice@alice (alice)

Hi,

There are no problems!

BR
Alice
?
```

Figure 27. A screenshot of Bob's mailbox.

It can be seen from the figure, that the TLSv1.2 encryption with a certain cipher suite was used to encrypt this particular email. The email transmission was also tested in the opposite direction, from Bob to Alice, by using a command

mail alice@example.com

and the received test results were consistent.

5 RESULTS AND DISCUSSION

This chapter introduces the current state, considers the results, which were found during the testing phase while encrypted email was tested, and discusses the importance of the additions and the results.

5.1 Current State

The methods of DNSSEC significantly improve the security and the reliability of the DNS. Even if DNSSEC has a few minor disadvantages, such as the complexity and the slowing effect to the overall processes, its benefits are so enormous, that it has gained a ground on the Internet as can be seen in Table 6, where the top 20 countries using DNSSEC on July 2013, are listed.

Table 6. Where is DNSSEC used? - The Top 20 [modified from 17, p. 12].

Rank	CC	Count	DNSSEC	Mixed	None	Country
1	SE	5,349	77.92	3.38	18.70	Sweden
2	SI	4,758	58.85	4.90	36.25	Slovenia
3	LU	652	43.87	6.90	49.23	Luxembourg
4	VN	26,665	38.28	4.04	57.69	Vietnam
5	FI	2,456	37.01	16.29	46.70	Finland
6	CZ	30,827	33.20	8.08	58.72	Czech Republic
7	CL	46,151	30.26	8.34	61.41	Chile
8	JM	1,545	28.22	3.11	68.67	Jamaica
9	IE	8,079	27.94	3.11	68.96	Ireland
10	BB	1,312	24.24	1.52	74.24	Barbados
11	ID	54,816	23.87	8.58	67.55	Indonesia
12	UA	26,399	21.65	12.75	65.60	Ukraine
13	ZA	2,969	21.15	9.36	69.48	South Africa
14	TR	49,498	18.06	2.10	79.84	Turkey
15	US	140,234	17.32	3.57	79.11	United States of America
16	EG	36,061	14.68	10.32	75.01	Egypt
17	GH	973	14.59	8.12	77.29	Ghana
18	AZ	7,409	14.55	30.34	55.11	Azerbaijan
19	BR	179,424	14.43	6.13	79.44	Brazil
20	PS	2,893	14.00	36.85	49.15	Occupied Palestinian Territory

In Table 6, the “DNSSEC” means, how many percent of clients appear to use the DNSSEC-validating resolvers. The “Mixed” means, how many percent of clients use a mix of DNSSEC-validating resolvers and non-validating resolvers. The “None” means, how

many percent of clients use the non-validating resolvers. Sweden is clearly the most advanced country on this market area.

On the website of the DNS and Domain Name Statistics and Tools (statdns) [18], there are listed all the 249 countries and their ccTLDs and only 90 of them (the ratio is 36.14%) have been signed by using DNSSEC.

Finland's ccTLD, .fi has been signed, but from the next level domains below it, only very few of them have been signed. There are, for example, only two banks in Finland, which have a signed domain name, Ålandsbanken and Danske Bank. This was validated in April 2014 by using Mozilla Firefox's plugin DNSSEC/TLSA Validator version 2.1.1, developed by CZ. NIC Labs. This plugin can also validate possible TLSA records within.

Finnish banks were asked via their own web site's contact forms about DNSSEC and in addition especially those banks, which do not use DNSSEC, were asked why is it not in use. First of all, of the banks, which already use DNSSEC, one did not answer at all and the other was not able to comment on its security solutions. From the banks, which do not use DNSSEC, a few did not answer. Some of those who answered, for example LähiTapiola, have considered DNSSEC, but have left it until further notice because in Finland, major security risks in DNS have not been detected. Also Osuuspankki have considered DNSSEC, but left it like LähiTapiola. Though, they have some other features for compensating DNSSEC, they were not able to comment further. For Aktia Bank, DNSSEC is a part of their broader security planning. Usage of DANE was also checked by using the DNSSEC/TLSA Validator, but none of the Finnish banks were using a TLSA records. All of these details are referring to the answers got via email.

5.2 Performance

Some of the key features related to the DNS queries and responses, introduced in Table 4, are gathered in Table 7. The results were considered from two different points of view: DNSSEC without DANE and DNSSEC with DANE. Their difference is announced and the ratio is calculated, which was a result of the values with DANE divided to the values without DANE.

Table 7. The comparison of the DNSSEC query with and without DANE.

	Without DANE	**With DANE**	**Difference**	**Ratio**
Number of packets	14	18	4	1.29
Number of bytes	3856 bytes	4904 bytes	1048 bytes	1.27
Elapsed time	108ms	117ms	9ms	1.07

It is seen from the table, that even if there were almost a 30% increase in the number of packets and the number of bytes, the most critical value, the elapsed time, only in-

creased by 7%. The elapsed times, during the several tests, were close to the measured value in Table 7. The test environment was simplified for measurements so it is expected, that slight variance may occur in the real network environment. The number of packets and bytes in a DNS query depends highly on the DNS's hierarchy and the elapsed time varies. Although, the TLSA record is stored in the name server, which administers the domain. The same delays would occur for the DNS responses as would for the TLSA responses, so the ratio should be relatively close to the measured value.

5.3 Certificate Packets

The Wireshark Packet Analyzer was used to capture TLS handshake packets, while connecting to various web sites. The web sites facebook.com, google.com and op.fi were used for testing. Of the captured packets, it was focused on the certificate packets and especially the lengths of including certificates and depths of the certificate chains. The captured certificate from mail.example.org was used as a reference. The captured packet is seen in Table 5 and the corresponding packet number is 40. Out of the total amount of 1497 bytes the share of the certificate is 970 bytes. The discovered certificate packet lengths and possible certificate chains are gathered in Table 8.

Table 8. Captured TLS handshake certificate packets.

	example.org	**facebook.com**	**google.com**	**op.fi**
Cert #1	Self-Signed Certificate **970 bytes**	facebook.com Certificate **1375 bytes**	google.com Certificate **1669 bytes**	op.fi Certificate **1469 bytes**
Cert #2	-	DigiCert High Assurance G2 CA-3 **1628 bytes**	Google Internet Authority G2 **1032 bytes**	VeriSign Class 3 Extended Validation SSL CA **1512 bytes**
Cert #3	-	-	GeoTrust Global CA **897 bytes**	VeriSign Class 3 Public Primary Certification **1236 bytes**
Total	**970 bytes**	**3003 bytes**	**3598 bytes**	**4217 bytes**

It is clearly seen in Table 8 that the self-signed certificate used as reference, which did not have a certificate chain, is the smallest packet and it therefore creates the smallest load to the network. The tested web sites had at least three times larger certificate packets. This correlates directly with the network traffic.

The total number of bytes, including the test phase's TLS handshake, is 2370 bytes. This was calculated from Table 5 starting from the packet number 37 and ending with the packet number 42. The share of the certificate is over 40 percent in this case, where the certificate is only as small as 970 bytes. Reducing 77 percent out of this

owned 40 percent share would mean a 30 percent reduction out of the entire certificate packet.

If the size of the certificate is 4217 bytes, the size of the certificate packet would be 4744 bytes in this case and the total number of bytes of the TLS handshake would be 5617 bytes. The share of the certificate would then already be a 75 percent out of the entire TLS handshake.

According to Table 7, the DANE implementation will increase the DNS traffic by 27 percent and 1048 bytes, but according to Table 8, it could instead reduce the size of the certificate packet by even 77 percent and 3247 bytes. The total number of bytes, when the certificate of op.fi is used (DANE is not in use), is 9473 bytes. The total number of bytes, when both, the self-signed certificate and DANE, are used, is 7274 bytes. The total reduction is 2199 bytes, which means a reduction of over 23 percent to the entire DNS and TLS handshake traffic.

5.4 Discussion

Implementing DANE is not complex. It only requires a TLSA record, which can easily be computed from the certificate. The calculated TLSA record is then copied to the DNS zone file and the DNS zone file is signed by using the DNSSEC methods. Because the DNS zone is necessary in order to sign again after every modification, it would be wise to schedule the future TLSA record rollovers at the same time with the DNSSEC key rollovers.

DANE limits the trust for only the parties who administrate the target DNS zone and its parent zones. DANE trusts the hierarchical chain of trust, where all the security is based on one trust anchor, for example, the root. There will not be a need anymore for hundreds of public Certification Authorities. This is emphasized with the connection of machine to machine (M2M) in the Internet of Things (IoT) as there will be no humans confirming untrusted certificates. Everything just has to work. Usually the case is, that the whole authentication process is missing.

For improving the existing performance and decreasing the delays, it could be a fine improvement, if the TLSA record was attached to the first DNS reply with an A or AAAA record, so there would be no need to request the TLSA records afterwards. If DNSSEC is in use, one option could be to send the DNS queries simultaneously. The first one queries the A or AAAA record and the second one queries the TLSA record. This would also reduce the delays since the scenarios, where there are no A or AAAA records, are unlikely.

6 CONCLUSIONS

The goal of this Master of Science Thesis was to study the DANE protocol and its major applications. DANE is a protocol, which uses TLSA records to associate a TLS certificate's data to the existing, hierarchical and reliable DNSSEC. From the M2M point of view, the main interest of this study was to research the DANE-validated TLS encryption between email servers.

DANE has been known to provide improved security and verification for TLS connections. As demonstrated earlier, DANE prevents the Man-in-the-Middle attack while negotiating an encrypted SMTP connection. One of the biggest advantages of this security improvement is that it allows the use of self-signed certificates, which means that the purpose of the third party CAs would significantly decrease. The self-signed certificates, DNSSEC and DANE combined enable the TLS encryption and authentication to all services, and especially to those parties who, in the past, have needed a CA to achieve just confidentiality.

In Chapter 4, the implementation of DANE, especially when used with encrypted email, was tested. The test results, presented in Chapter 5, proved that if there is already an existing DNSSEC, the implementation of DANE is not complex and does not require advanced expertise. The addition of DANE does not increase the delays or the number of bytes in DNS queries significantly. Actually, the addition of DANE could even decrease the total number of bytes during the TLS handshake, when a self-signed certificate is used instead of a certificate issued by a third party CA. The reduction in the number of bytes, due to the use of a self-signed certificate, will compensate the increase in the number of bytes in DNS traffic. Overall, the process could require even 23 percent less bytes.

The study has shown, that DANE is a recommended technique to put into service immediately after the organization's DNSSEC launch. DANE provides good extra security for encrypted web sites and email transmissions, which TLS and DNSSEC alone can not offer.

REFERENCES

[1] Dierks, T., Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.2*, *IETF RFC 5246*, August 2008. Available at: http://tools.ietf.org/html/rfc5246.

[2] Cisco Security Intelligence Operations, *DNS Best Practices, Network Protections, and Attack Identification*, [accessed on 19.10.2013], available at: http://www.cisco.com/web/about/security/intelligence/dns-bcp.html.

[3] Australian Government Intelligence Agency – Australian Signals Directorate (ASD), *Domain Name System (DNS) Security Strategies*, August 2012, [accessed on 21.10.2013], available at: http://www.asd.gov.au/publications/csocprotect/dns_security.htm.

[4] Mockapetris, P., *Domain Names – Implementation and Specification, IETF RFC 1035*, November 1987. Available at: http://tools.ietf.org/html/rfc1035.

[5] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., *DNS Security Introduction and Requirements, IETF RFC 4033*, March 2005. Available at: http://tools.ietf.org/html/rfc4033.

[6] DNSSEC: *DNS Security Extensions, Securing the Domain Name System*, [accessed on 1.11.2013], available at: http://www.dnssec.net/.

[7] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., *Resource Records for the DNS Security Extensions, IETF RFC 4034*, March 2005. Available at: http://tools.ietf.org/html/rfc4034.

[8] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., *Protocol Modifications for the DNS Security Extensions, IETF RFC 4035*, March 2005. Available at: http://tools.ietf.org/html/rfc4035.

[9] Conrad, D., *Indicating Resolver Support of DNSSEC, IETF RFC 3225*, December 2001. Available at: http://tools.ietf.org/html/rfc3225.

[10] Vixie, P., *Extension Mechanisms for DNS (EDNS0), IETF RFC 2671*, August 1999. Available at: http://tools.ietf.org/html/rfc2671.

[11] Hopkins, A., Borne, J. C., *Helping Secure the Internet with DNSSEC*, September 2010, [accessed on 18.12.2013]. Available at: http://www.educause.edu/ero/article/helping-secure-internet-dnssec.

[12] Galvin, J., *Why DNSSEC?*, November 2013, [accessed on 15.1.2014]. Available at: http://www.slideshare.net/Deploy360/ion-toronto-why-implement-dnssec.

[13] Mohan, R., *Five Strategies for Flawless DNSSEC Key Management and Rollovers*, July 2010, [accessed on 15.1.2014]. Available at: http://www.securityweek.com/five-strategies-flawless-dnssec-key-management-and-rollover.

[14] Kolkman, O., RIPE NCC PROJECT, *DNSSEC Key Management and Zone Signing*, [accessed on 15.1.2014]. Available at: http://www.ripe.net/data-tools/projects/archive/disi/dnssec-key-management-and-zone-signing.

[15] Kolkman, O., Mekking, W., Gieben, R., *DNSSEC Operational Practices, Version 2, IETF RFC 6781*, December 2012. Available at: http://tools.ietf.org/html/rfc6781.

[16] EDUCAUSE, *7 Things You Should Know About DNSSEC*, January 2010, [accessed on 22.1.2014]. Available at: https://net.educause.edu/ir/library/pdf/EST1001.pdf.

[17] Huston, G., Michaelson, G., *Measuring DNSSEC use*, July 2013, [accessed on 23.1.2014]. Available at: http://iepg.org/2013-07-ietf87/2013-07-28-dnssec.pdf.

[18] Website of statdns, *DNS and Domain Name Statistics and Tools – List of ccTLDs*, [accessed on 23.1.2014]. Available at: http://www.statdns.com/cctlds/.

[19] Mills, E., McCullagh, D., *CNET News – Google, Yahoo, Skype targeted in attack linked to Iran*, March 2011, [accessed on 23.1.2014]. Available at: http://news.cnet.com/8301-31921_3-20046340-281.html.

[20] Mills, E., *CNET News - Fraudulent Google certificate points to Internet attacks*, August 2011, [accessed on 23.1.2014]. Available at: http://news.cnet.com/8301-27080_3-20098894-245/fraudulent-google-certificate-points-to-internet-attack/.

[21] Gieben, R., *Chain of Trust – The parent-child and keyholder-keysigner relations and their communication in DNSSEC*, January 2013, [accessed on 30.1.2014]. Available at: http://www.nlnetlabs.nl/downloads/publications/CSI-report.pdf.

[22] Hoffman, P., Schlyter, J., *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TSLA, IETF RFC 6698*, August 2012. Available at: http://tools.ietf.org/html/rfc6698.

[23] Cooper, D., et al., *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, IETF RFC 5280*, May 2008. Available at: http://tools.ietf.org/html/rfc5280.

[24] Eastlake 3rd, D., Hansen, T., *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF), IETF RFC 6234*, May 2011. Available at: http://tools.ietf.org/html/rfc6234.

[25] Barnes, R. L., *The IETF Journal, DANE: Taking TLS Authentication to the Next Level Using DNSSEC*, October 2011, [accessed on 31.1.2014]. Available at: http://www.internetsociety.org/articles/dane-taking-tls-authentication-next-level-using-dnssec.

[26] Gudmundsson, O., *Adding acronyms to simplify DANE conversations draft-ietf-dane-registry-acronyms-04, IETF Internet-Draft*, February 2014. Available at: http://tools.ietf.org/html/draft-ietf-dane-registry-acronyms-04.

[27] Ramsdell, B., Turner, S., *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification, IETF RFC 5751*, January 2010. Available at: http://tools.ietf.org/html/rfc5751.

[28] Hoffman, P., Schlyter, J., *Using Secure DNS to Associate Certificates with Domain Names For S/MIME draft-ietf-dane-smime-06, IETF Internet-Draft*, February 2014. Available at: http://tools.ietf.org/html/draft-ietf-dane-smime-06.

[29] York, D., Internet Society Deploy360 Programme, *The DANE Protocol – DNS-Based Authentication of Named Entities*, October 2012, [accessed on 27.2.2014]. Available at: http://www.internetsociety.org/deploy360/resources/dane/.

[30] Kumari, W., York, D., Internet Society Deploy360 Programme, *The DANE Protocol: What It Is And Why It Matters (including DNSSEC, SSL/TLS) – video interview*, August 2012, [accessed on 27.2.2014]. Available at: http://www.internetsociety.org/deploy360/resources/dane/ or at: http://www.youtube.com/watch?v=emDxUQl1NvA.

[31] Barnes, R. L., Internet Society Deploy360 Programme, *DANE: Taking TLS Authentication to the Next Level using DNSSEC*, October 2011, [accessed on 27.2.2014]. Available at: http://www.internetsociety.org/articles/dane-taking-tls-authentication-next-level-using-dnssec.

[32] Klensin, J., *Simple Mail Transfer Protocol, IETF RFC 5321*, October 2008. Available at: http://tools.ietf.org/html/rfc5321.

[33] Dukhovni, V., Hardaker, W., *SMTP security via opportunistic DANE TLS draft-ietf-dane-smtp-with-dane-07, IETF Internet-Draft,* February 2014. Available at: http://tools.ietf.org/html/draft-ietf-dane-smtp-with-dane-07.

[34] Dukhovni, V., Hardaker, W., *DANE for SMTP – IETF 87, Berlin*, July 2013, [accessed on 14.3.2014]. Available at: http://www.ietf.org/proceedings/87/slides/slides-87-dane-2.pdf.

[35] Hoffman, P., *SMTP Service Extension for Secure SMTP over Transport Layer Security, IETF RCF 3207*, February 2002. Available at: http://tools.ietf.org/html/rfc3207.

[36] NLnet Labs, *Manual for NSD configuration*, March 2014, [accessed on 14.4.2014]. Available at: http://www.nlnetlabs.nl/projects/nsd/nsd.conf.5.html.

[37] NLnet Labs, *Tutorial for Unbound Library*, March 2014, [accessed on 14.4.2014]. Available at: http://www.unbound.net/documentation/unbound.conf.html.

[38] *Ubuntu Official Documentation, Ubuntu 14.04 - Ubuntu Server Guide - Email Services - Postfix,* [accessed on 14.4.2014]. Available at: https://help.ubuntu.com/14.04/serverguide/postfix.html.

[39] *Ubuntu Official Documentation, Ubuntu 14.04 - Ubuntu Server Guide - Security - Certificates,* [accessed on 14.4.2014]. Available at: https://help.ubuntu.com/14.04/serverguide/certificates-and-security.html.

[40] *The Postfix Home Page - Documentation: TLS Encryption and Authentication*, [accessed on 15.4.2014]. Available at: http://www.postfix.org/TLS_README.html.